Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

QR Factorization of Block Low-rank Matrices with Weak Admissibility Condition

Akihiro Ida[1,a)] Hiroshi Nakashima[2] Tasuku Hiraishi[2] Ichitaro Yamazaki[3]
Rio Yokota[4] Takeshi Iwashita[5]

Received: April 5, 2019, Accepted: August 19, 2019

Abstract: The QR factorization of a matrix is a fundamental operation in linear algebra and it is widely utilized in scientific simulations. Although the QR factorization requires a memory storage of $O(N^2)$ and the computational cost of $O(N^3)$, they can be reduced if the matrix could be approximated using low-rank structured matrices, such as hierarchical matrices ($\mathcal{H}$-matrices). In this paper, we study the QR factorization based on the block low-rank (BLR-) matrix representation, which is a simplified variant of the $\mathcal{H}$-matrix. We demonstrate how the BLR block size should be defined in such matrices and confirm that the memory and computational complexities of the BLR are $O(N^{1.5})$ and $O(N^2)$ when using an appropriate block size. Furthermore, using the simple structure of BLR-matrices, we also present a parallel algorithm for the QR factorization of BLR-matrices on distributed memory systems. In numerical experiments performed, we observe that the accuracy of the QR factorization is controllable by the accuracy of the BLR-matrix approximation of the original matrix. Finally, we verify that our algorithm enables us to perform the QR factorization of matrices of several hundred thousand N within a reasonable amount of time using moderate computer resources.

Keywords: parallel computing, low-rank approximation, block low rank matrices, QR factorization

1. Introduction

Dense matrices appear in many scientific simulations, for example, in the discretization of integral equations. Dense matrices appear also as the Schur complements during the LU factorization of sparse matrices obtained from differential equations, and as the orthogonal matrices derived from the QR factorization of sparse matrices. The dense square matrix requires the memory usage of $O(N^2)$ and the computational cost of $O(N^3)$ for arithmetic operations such as matrix–matrix multiplication, matrix inversion, and LU and QR decompositions, where N is the number of rows (columns) in the matrix. These requirements prohibit large-scale simulations.

An approximation technique for the dense matrices can be used to reduce the computational cost and memory consumption. The low-rank structured matrices that are represented by hierarchical ($\mathcal{H}$-)matrices [1] are approximation methods based on low-rank approximation (LRA) methods. For example, $\mathcal{H}$-matrices obtain a computational complexity of $O(N \log^2 N)$ for LU factorization with a memory complexity of $O(N \log N)$ [1], [2].

Parallel computing on a distributed-memory computing system offers another solution to performing large-scale simulations. To achieve good parallel scalability, we must balance the workload and construct an efficient communication pattern among the parallel processing units. Unfortunately, the complicated structure of $\mathcal{H}$-matrices like shown in **Fig. 1** (A) prevents one from satisfying these requirements simultaneously. However, simplification of the structure could solve this problem. Block low-rank (BLR-) matrices [3] are regarded as special cases of $\mathcal{H}$-matrices with simplified structures. Although the simplification of structures results in an increase in memory usage and computational costs, it is shown in Ref. [4] that matrix-vector multiplications based on BLR-matrices are significantly faster than an $\mathcal{H}$-matrix version for a large number of processes.

In this paper, we consider QR factorization based on the BLR-matrices with a weak admissibility condition, which has the simplest format among all low-rank structured matrices. The admissibility condition determines whether each block of the low-rank structured matrices is represented by a low-rank matrix or a dense matrix. In the case of the weak admissibility condition [5], all the submatrices except those on the diagonal become low-rank. BLR-matrices are characterized by a simple, non-hierarchical, and low-rank format based on lattice structures, as shown in Fig. 1 (B). These lattice structures are similar to the format of a block divided dense matrix and, thus, allow the use of existing algorithms for dense matrices. Due to the advantages above, BLR-matrices have recently been applied to realize complex arithmetic functions in parallel processing, such as the approximation of Schur complements in the multi-frontal LU factorization [3] and the approximation of covariance matrices in maximum likelihood estimation for climate modelling applications [6]. If $\mathcal{H}$-matrices could be applied to these applications, a more efficient approximation would be performed. However, except for simple oper-

[1] The University of Tokyo, Bunkyo, Tokyo 113–8658, Japan
[2] Kyoto University, Kyoto 606–8501, Japan
[3] Sandia National Laboratories, Albuquerque, New Mexico 87185, USA
[4] Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[5] Hokkaido University, Sapporo, Hokkaido 060–0811, Japan
[a)] ida@cc.u-tokyo.ac.jp

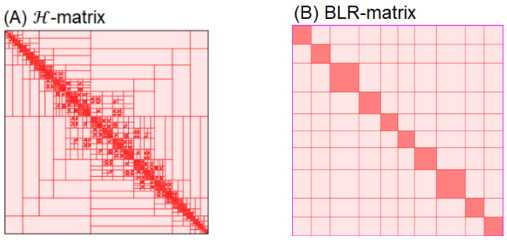


Fig. 1 An $\mathcal{H}$-matrix structure (A) and its conversion to a BLR-matrix with a weak admissibility condition (B). Blocks painted in deep red show dense submatrices, and the off-diagonal blocks in light red indicate low-rank submatrices.

ations such as the matrix-vector multiplication, it is hard to efficiently implement arithmetic of $\mathcal{H}$-matrices, especially in parallel processing.

The rest of this paper is structured as follows. In Section 2, we discuss the related works of this study. In Section 3, the modified block Gram Schmidt (MBGS) algorithm is introduced as a QR factorization algorithm for usual dense matrices. We propose an MBGS algorithm for BLR-matrices with a weak admissibility condition in Section 4. How to determine the BLR block size is also discussed, and the memory and computational complexities are estimated when using the appropriate block size. Furthermore, using the simple structure of BLR-matrices, a parallel algorithm for the QR factorization of BLR-matrices is presented. In Section 5, some numerical experiments of our proposed MBGS algorithm for BLR- matrices are presented. The last section contains conclusions and future work.

2. Related Work

Most low-rank structured matrices can be regarded as a special case of $\mathcal{H}$-matrices proposed by Hackbusch [1]. In the studies of low-rank structured matrices, arithmetic functions (e.g., matrix generation, matrix-vector multiplication, matrix-matrix multiplication, matrix inversion, and LU factorization) are often discussed. However, no library supports the QR factorization of low-rank structured matrices, and only a few serial implementations of the QR factorization of $\mathcal{H}$-matrices ($\mathcal{H}$-QR) have been reported in the literature (see, e.g., Ref. [7]). For the $\mathcal{H}$-QR, functions related to the summation and multiplication of low-rank submatrices are required. However, due to the complex structures of $\mathcal{H}$-matrices, realizing these arithmetic functions on distributed-memory computing systems is complicated. Even if the implementation of $\mathcal{H}$-QR is realized, parallel efficiency would be difficult to achieve.

Although the $\mathcal{H}$-matrices achieve a good asymptotic complexity by exploiting their adaptive (recursive) block structure, they are difficult to implement, particularly on distributed memory systems. There are a few existing works on the parallelization of $\mathcal{H}$-matrices (see, e.g., Refs. [8], [9]). However, the parallel scalability of these implementations is not adequate. To improve the parallel scalability and the convenience of matrix operations for $\mathcal{H}$-matrices, some formats with a simpler structure have been proposed in the literature, such as hierarchically semi-separable matrices [10], hierarchically off-diagonal low-rank matrices [11], block low-rank (BLR) matrices [5], and lattice $\mathcal{H}$-matrices [12]. BLR-matrices with a weak admissibility condition is one of the simplest formats among the structured low-rank matrices, whose structures are similar to the format of a block divided dense matrix.

The QR factorization of a matrix is a fundamental operation in linear algebra. Various efficient algorithms for block divided dense matrices have been proposed in the literature [13], [14], [15] and some of them are included in popular libraries, such as LAPACK and ScaLAPACK [16]. A parallel algorithm has been proposed for factorizing the block divided dense matrices in Ref. [17]. The current work extends the idea to factorizing BLR-matrices. To the best of our knowledge, this is the first study on the QR factorization of BLR-matrices. The parallel algorithm of low-rank structured matrices is implemented here for the first time.

3. QR Factorization of a Usual Dense Matrix

For preparation to discuss the QR factorization based on BLR-matrices, we here recall the QR factorization of a usual dense matrix. A given matrix $A \in \mathbb{R}^{m\times n}$ can be factorized as

$$A := QR, \tag{1}$$

where $Q \in \mathbb{R}^{m\times n}$ is an orthogonal matrix and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. To simplify the notation, hereafter, we assume square matrices, i.e., $m = n = N$.

Various algorithms have been proposed for the QR factorization, and the block-divided technique is adapted in some algorithms to efficiently perform the factorization on parallel systems. In the technique, the matrices are divided into $N_b \times N_b$ blocks (submatrices) as

$$X := \{X_{ij} \mid 1 \le i, j \le N_b\}, \quad X \in \{A, Q, R\}. \tag{2}$$

The matrix is subdivided into a lattice structure like shown in Fig. 1 (B). By performing operations block-by-block, we can utilize efficient BLAS-3 functions on each process and construct an efficient communication pattern between the processes.

Among the block-partitioned QR factorization algorithms, we focus on the modified block Gram Schmidt (MBGS) algorithm [18], whose procedure is shown in **Fig. 2**. In the case where the matrix A to be factorized is dense, the computational complexity of the MBGS algorithm is $O(N^3)$ regardless of the block size. **Table 1** shows the step-by-step breakdown of the complexity assuming that all blocks are square and have equal size l ($:= N/N_b$) to simplify the estimation. We observe that the leading factors are steps 4 and 5 in Fig. 2, and they require $O(N^3)$ operations regardless of the block size l.

4. Our Proposal QR Factorization Based on a BLR-matrix Representation

4.1 Introduction to BLR-matrices

Assuming that all the off-diagonal submatrices on the block-divided matrix A in Eq. (2) are represented in the form of a low-rank approximation (LRA), while diagonal submatrices remain dense (see **Fig. 3**), we formally obtain the simplest format of low-rank structured matrices, which is categorized as a BLR-matrix with a weak admissibility condition. A BLR-matrix $\tilde{A}$, which consists of submatrices $\tilde{A}_{ij}$ corresponding to the original dense

```
Input : A with N_b × N_b blocks
Output : Q, R with N_b × N_b blocks
         such that QR=A
1: For   j = 1,2,⋯,N_b  do
2:     [Q_{*,j}, R_{j,j}] ≔ TSQR(A_{*,j})
3:       For   k = j+1, j+2, ⋯, N_b  do
4:           R_{j,k} ≔ Q^T_{*,j} A_{*,k}
5:           A_{*,k} ≔ A_{*,k} − Q_{*,j}  R_{j,k}
6:       End do
7: End do
```

Fig. 2 Modified Block Gram Schmidt (MBGS) algorithm, where N_b is the number of blocks in a column (or row) of matrices A, Q, and R. The subscripts of the matrices denote the row and column indices of the lattice. The symbol * represents all blocks, e.g., $A_{*,j}$ is the j-th block column which is a tall-skinny matrix. The function TSQR($A_{*,j}$) returns the QR factorization of the tall-skinny matrix $A_{*,j}$ such that $Q_{*,j}^T Q_{*,j} = I$ and $Q_{*,j}, R_{j,j} = A_{*,j}$ where $Q_{*,j} \in \mathbb{R}^{N\times l}$ and $R_{j,j} \in \mathbb{R}^{l\times l}$.

Table 1 Arithmetic complexity of the QR factorization of a usual dense matrix. In the table, "# of calls" denotes the number of times a given function is called, "1time complexity" is the arithmetic complexity of each function call, and "Total complexity" is the product of these two.

Function	# of calls	1time complexity	Total complexity
$\mathrm{TSQR}(A_{*,j})$	$O(N/l(:= N_b))$	$O(l^2N)$	$O(lN^2)$
$Q_{*,j}^T A_{*,k}$	$O((N/l)^2)$	$O(l^2N)$	$O(N^3)$
$A_{*,k} - Q_{*,j}\, R_{j,k}$	$O((N/l)^2)$	$O(l^2N)$	$O(N^3)$

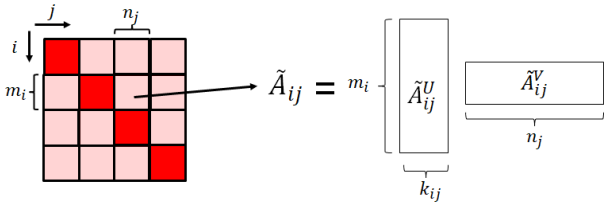


Fig. 3 BLR-matrix and low rank representation of a sub-matrix. An $m_i \times n_j$ matrix $\tilde{A}_{ij}$ with rank k_{ij} is represented in the form of an LRA, which is a multiplication of an $m_i \times k_{ij}$ matrix $\tilde{A}^U_{ij}$ and an $k_{ij} \times n_j$ matrix $\tilde{A}^V_{ij}$: $\tilde{A}_{ij} = \tilde{A}^U_{ij}\tilde{A}^V_{ij}$.

submatrices $A_{ij} \in \mathbb{R}^{m_i \times n_j}$, is defined as

$$\tilde{A} := \left\{ \tilde{A}_{ij} = \begin{cases} \tilde{A}^U_{ij}\tilde{A}^V_{ij} & (i \neq j) \\ A_{ij} & (i = j) \end{cases} \middle| \; 1 \leq i, j \leq N_b \right\}, \tag{3}$$

where $\tilde{A}^U_{ij} \in \mathbb{R}^{m_i \times k_{ij}}$, $\tilde{A}^V_{ij} \in \mathbb{R}^{k_{ij} \times n_j}$, and $k_{ij} \in \mathbb{N}$ are regarded as the rank of the submatrix $\tilde{A}_{ij}$. We suppose that m_i and n_j are $O(l)$. If $k_{ij} \ll \min(m_i, n_j)$ for most of the submatrices, the memory usage of the BLR-matrix $\tilde{A}$ is much smaller than the dense matrix. For a given error tolerance $\varepsilon \in \mathbb{R}_{>0}$, we assume the following expression is satisfied: $\|A_{ij} - \tilde{A}_{ij}\|_F / \|\tilde{A}_{ij}\|_F \leqq \varepsilon$, where $\|\cdot\|_F$ denotes the Frobenius norm. Then, for the approximated matrix $\tilde{A}$, we can prove $\|A - \tilde{A}\|_F / \|\tilde{A}\|_F \leqq \varepsilon$ [19]. Therefore, the error tolerance for the LRA method determines the accuracy of the approximated entire matrix.

Memory usage of the BLR-matrix $\tilde{A}$ depends on the block sizes. When assuming a uniform block size l and rank $k(\ll l)$ for all low-rank submatrices, the dependency of memory usage on the block size l is given by the following function $f(l)$:

$$f(l) = lN + 2kN\left(\frac{N}{l} - 1\right). \tag{4}$$

At point $l_{\min} = \sqrt{2kN}$, $f(l)$ assumes the minimum value

$$f(l_{\min}) = 2\sqrt{2kN^3} - 2kN. \tag{5}$$

Therefore, we find that the memory complexity of BLR-matrices is $\tilde{O}(N^{1.5})$, where the symbol $\tilde{O}$ assumes $k \ll l$ and hence a complexity order that omit the rank k. Hereafter in this section 4, we use the symbol $\tilde{O}$ to differ from the symbol O for a complexity order without any influence by the rank k. The memory complexity of BLR-matrices $\tilde{O}(N^{1.5})$ is reduced from that of dense matrices $O(N^2)$.

4.2 Application of the MBGS Algorithm to BLR-matrices

Using the MBGS algorithm in Fig. 2, we consider the QR factorization of the BLR-matrix $\tilde{A}$ in Eq. (3). As with the case of QR factorization of a dense matrix, we perform the operations block-by-block. The difference is submatrices in the off-diagonal blocks are formatted by the low-rank representation. We enforce the following two conditions on the computed matrices $\tilde{Q}$ and $\tilde{R}$:

C1: The matrices $\tilde{Q}$ and $\tilde{R}$ have the same lattice structure as $\tilde{A}$.

C2: The off-diagonal submatrices $\tilde{Q}_{ij}$ and $\tilde{R}_{ij}$ are formatted by the same rank k_{ij} representation as $\tilde{A}_{ij}$.

To satisfy these conditions, the following four arithmetic operations with low-rank submatrices are required in addition to the usual multiplication and summation of dense matrices.

LA1: (low-rank) + (low-rank) = (low-rank)

LA2: (low-rank) × (low-rank) = (low-rank)

LA3: (low-rank) + (dense) = (dense)

LA4: (low-rank) × (dense) = (low-rank)

The summation of the two low-rank submatrices (LA1) increases the rank of the resulting matrix (**Fig. 4**), and the condition C2 cannot be satisfied without approximating the resulting matrix. To enforce the condition, we further recompress the resulting matrix to reduce the rank. Although we have a choice of when to perform the approximation, in this study, it is done every time the summation of the two low-rank submatrices occurs. For the approximated summation, we employ the "rounded addition" method proposed in Ref. [20]. In this method, the increased rank is reduced by the efficient use of the TSQR and singular value decomposition (SVD) as shown in **Fig. 5**. When the dimension and the rank of the two low-rank submatrices are denoted by l and k, the complexity of the rounded addition is $\tilde{O}(l)$. Under these conditions C1 and C2, the multiplication $\tilde{Q}\tilde{R}$ would not be equal to $\tilde{A}$, but an approximation: $\tilde{A} \approx \tilde{Q}\tilde{R}$.

For a BLR-matrix with the weak admissibility condition, at least one of the matrices needed for the matrix operations on lines 4 and 5 of MBGS (Fig. 2) is a low-rank matrix. Hence, we can use the low-rank arithmetic operations LA1-4 above (there is no operation with two dense matrices that would require $O(l^3)$ arithmetic operations). The arithmetic LA4 is needed two times on line 4, and on line 5, we use LA3 and LA4 once. When one of the blocks is a dense matrix, the computational complexity of these operations is $\tilde{O}(l^2)$. On the other hand, if both blocks were

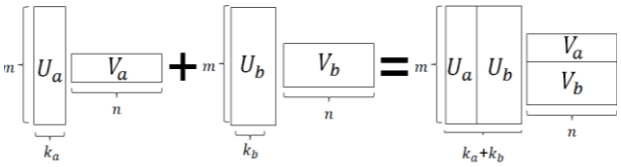


Fig. 4 Summation of two submatrices with a low-rank representation. For a matrix a with a rank k_a and a matrix b with a rank k_b, the rank of the matrix "$a + b$" is equal to "$k_a + k_b$".

Input : $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{k\times n}$ composing $A \coloneqq UV$
Output : $U' \in \mathbb{R}^{m\times k'}$, $V' \in \mathbb{R}^{k'\times n}$
such that $U'V'$ approximates UV
1: $[Q^U, R^U] \coloneqq \mathrm{TSQR}(U)$, $[Q^V, R^V] \coloneqq \mathrm{TSQR}(V)$
2: $[\mu, \sigma, \upsilon] \coloneqq \mathrm{SVD}(R^U R^V)$
3: Truncate σ at k' : $\sigma \in \mathbb{R}^{k\times k} \longrightarrow \sigma' \in \mathbb{R}^{k'\times k'}$
4: Set U' as the first k' columns of $Q^U\mu\sigma'$
5: Set V' as the first k' rows of υQ^V

Fig. 5 Rank reduction algorithm for matrices with a low-rank representation, where $(Q^U\mu)\sigma(\upsilon Q^V)$ is the SVD of UV and the corresponding truncated SVD with the k' largest singular values gives the best rank-k' approximation of UV.

low-rank matrices, we only require the computational complexity of $\tilde{O}(l)$. Among a total of $O(N/l)$ blocks within each block column, only one block is dense. Hence, for orthogonalizing the k-th block column $\tilde{A}_{*,k}$ against $\tilde{A}_{*,j}$, the complexity of these operations on lines 4 and 5 is $\tilde{O}(N + l^2)$ at each j-th step.

The remaining function is TSQR for a block column $\tilde{A}_{*,j}$ in line 2. For this, we follow the method proposed in Ref. [7]. The j-th block column of the BLR-matrix $\tilde{A}$ in Eq. (3) consists of a dense submatrix A_{jj} and $N_b - 1$ low-rank submatrices. We represent the dense submatrix by a low-rank representation, i.e., $\tilde{A}_{jj} = \tilde{A}^U_{jj}\tilde{A}^V_{jj}$, where $\tilde{A}^U_{jj} := I$ and $\tilde{A}^V_{jj} := A_{jj}$. Then, we represent the j-th block column by

$$\tilde{A}_{*,j} = [\tilde{A}^U_{1j}\tilde{A}^V_{1j}, \tilde{A}^U_{2j}\tilde{A}^V_{2j}, \cdots, \tilde{A}^U_{N_bj}\tilde{A}^V_{N_bj}]^T. \tag{6}$$

For each tall-skinny matrix $\tilde{A}^U_{ij}$, we perform the QR factorization as $\tilde{A}^U_{ij} = \tilde{Q}^U_{ij}\tilde{R}^U_{ij}$ whose computational complexity is $O(k_{ij}l^2)$. Therefore, the complexity of computing the QR factorization of all blocks $\tilde{A}^U_{ij}$ $(i = 1 \cdots N_b)$ is $O(Nlk_{ij})$ i.e., $\tilde{O}(Nl)$. The block column $\tilde{A}_{*,j}$ is decomposed as

$$\tilde{A}_{*,j} = \begin{bmatrix} \tilde{Q}^U_{1j} & 0 & \cdots & 0 \\ 0 & \tilde{Q}^U_{2j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \tilde{Q}^U_{N_bj} \end{bmatrix} B_{*,j}, \tag{7}$$

where

$$B_{*,j} := \begin{bmatrix} \tilde{R}^U_{1j}\tilde{A}^V_{1j} \\ \tilde{R}^U_{2j}\tilde{A}^V_{2j} \\ \vdots \\ \tilde{R}^U_{N_bj}\tilde{A}^V_{N_bj} \end{bmatrix}. \tag{8}$$

The number of rows of each block $B_{i,j} := \tilde{R}^U_{ij}\tilde{A}^V_{ij}$ is a small number of rank $k_{ij}(\ll l)$, except for $B_{j,j}$ whose number of rows is the block size, i.e., $B_{*j} \in \mathbb{R}^{(m_j+\sum_i k_{ij})\times n_j}$. Therefore, we can carry out the QR factorization of B_{*j} with $\tilde{O}(l^3)$ as $B_{*j} = \tilde{Q}^B_j\tilde{R}^B_j$ where $\tilde{Q}^B_j \in \mathbb{R}^{(m_j+\sum_i k_{ij})\times n_j}$ and $\tilde{R}^B_j \in \mathbb{R}^{n_j\times n_j}$. By slicing the matrix $\tilde{Q}^B_j$, it can be formatted in the same way as the right hand side of Eq. (8). Finally, we calculate $[\tilde{Q}_{*j}, \tilde{R}_{jj}] := \mathrm{TSQR}(\tilde{A}_{*j})$ where $\tilde{R}_{jj} := \tilde{R}^B_j$ and

$$\tilde{Q}_{*,j} = \begin{bmatrix} \tilde{Q}^U_{1j}\tilde{Q}^B_{1j} \\ \tilde{Q}^U_{2j}\tilde{Q}^B_{1j} \\ \vdots \\ \tilde{Q}^U_{N_bj}\tilde{Q}^B_{N_bj} \end{bmatrix}. \tag{9}$$

The computational complexity of $\mathrm{TSQR}(\tilde{A}_{*,j})$ is $\tilde{O}(l^3)$. The leading factor is the QR factorization of $B_{i,j}$ in Eq. (8).

In the case of a BLR-matrix, the computational complexity of the MBGS algorithm depends on the block sizes. Assuming that all the blocks are square and have the same size l, the breakdown of the complexity is shown in **Table 2**. The complexity becomes $O(N^3)$ when $l = 1$ and $l = N$ which correspond to a dense matrix. For $l \propto \sqrt{N}$, the complexity takes its minimum $\tilde{O}(N^2)$.

Table 2 Arithmetic complexity of the QR factorization of a BLR-matrix based on the MBGS algorithm. In the table, "# of calls" denotes the number of times a given function is called, "1time complexity" is the arithmetic complexity of each function call, and "Total complexity" is the product of these two.

Function	# of calls	1time complexity	Total complexity
$\mathrm{TSQR}(A_{*,j})$	$O(N/l)$	$\tilde{O}(l^3)$	$\tilde{O}(Nl^2)$
$Q^T_{*,j}A_{*,k}$	$O((N/l)^2)$	$\tilde{O}(N + l^2)$	$\tilde{O}(N^3/l^2 + N^2)$
$A_{*,k} - Q_{*,j}R_{j,k}$	$O((N/l)^2)$	$\tilde{O}(N + l^2)$	$\tilde{O}(N^3/l^2 + N^2)$

4.3 Parallel MBGS Algorithm for BLR-matrices

For the parallelization of the MBGS algorithm in Fig. 2 using the BLR-matrix representation, we use a hybrid MPI+OpenMP programming model, and carefully consider the assignment of tasks to MPI processes.

Although most operations are performed block-by-block, the QR factorization of B_{*j} in Eq. (7) needs all the information in the j-th block column. To simplify the implementation, we divide the whole matrix into block columns, and assign a set of block columns to an MPI process. Although we sometimes assumed square blocks with a uniform block size l and rank k for all low-rank submatrices in the previous sections to estimate the memory usage of BLR-matrices and the computational cost of the MBGS, blocks actually have a rectangular shape and low-rank submatrices differ in rank. To balance the load among MPI processes, we adopt a block cyclic assignment strategy in the column direction. As a result of this, the p-th MPI process is assigned a set of submatrices:

$$\tilde{A}^p := \{\tilde{A}_{*j} \mid \mathrm{Mod}(j-1, N_p) = p\}, \tag{10}$$

where N_p is the number of MPI processes. In each MPI process, the QR factorization required for the derivation of Eq. (7) from Eq. (6) in the step 2 (TSQR) in Fig. 2 is carried out by using the `dgeqrf` subroutine provided in the LAPACK library [16] which are threaded with the OpenMP technology. The matrix–matrix

```
Input : Ã^p with N_b × N_bp blocks assigned
        to p^th MPI process
Output : Q̃^p , R̃^p with N_b × N_bp blocks
1: For j = 1,2,⋯,N_b do
2:   If(p = Mod(j−1, N_p)) then
3:       [Q̃_*j, R̃_jj] ≔ TSQR(Ã_*j) using OpenMP
4:       Broadcast(Q̃_*j, R̃_jj)
5:   End if
6:   Sync
7:   For  k = j + 1, j + 2, ⋯, N_b do
8:       If (p = Mod(k − 1, N_p)) then
9:          R̃_jk ≔ Q̃^T_*j Ã_*k using OpenMP
10:      End if
11:  End do
12:  For all Ã_ik ∈ {Ã^p ∩ k > j} do using
     OpenMP schedule(dynamic)
13:         Ã_ik ≔ Ã_ik − Q̃_ij R̃_jk
14:  End do
15: End do
```

Fig. 6 MBGS on the p^{th} MPI process. Arithmetic function on $N_b \times N_{bp}$ blocks is performed, where N_{bp} denotes the number of block columns assigned to the p^{th} MPI process.

multiplications in the step 4 ($Q^T_{*,j}A_{*,k}$) are performed by using the `dgemm` subroutine in the LAPACK. Furthermore, the step 5 is performed block-by-block using OpenMP dynamic scheduling. **Figure 6** shows the parallel MBGS algorithm for BLR-matrices under the assignment (10).

5. Numerical Experiments

5.1 Implementation and Computation Environment

In this subsection, we discuss the performance of the MBGS algorithm for BLR-matrices. For the algorithm, we re-implemented functions of the $\mathcal{H}$ACApK library [21], which provides the functions for the generation of low-rank structured matrices that work on distributed-memory systems using MPI. For the common parts of these low-rank structured matrices, such as LRA or submatrix-vector multiplication, the functions in the $\mathcal{H}$ACApK library were used in the new implementation where possible. Throughout the numerical experiments, we use the adaptive cross approximation (ACA) [22] to perform the LRA.

As a test problem, we have selected the static electric field analyses described in the next subsection. All calculations were carried out using an SMP cluster system, which is equipped with Intel(R) Xeon(R) E5-2680 v2 (10core × 2 sockets/node) and 32 GB DDR3 memory on a node. For the interconnect, a Fat-Tree with Full-bisection bandwidth using InfiniBand FDR×2 is used, which has a link throughput of 6.8 GB/s. We used the Intel Fortran compiler with the –O3 optimization option and the Intel MPI and MKL libraries. We adopted the hybrid MPI+OpenMP programming model.

5.2 BLR-matrices Derived from BEM Analyses

We use an electrostatic field problem to get examples of target matrices [23] and assume a spherical perfect conductor. It is set in a uniform electric field in the z-direction in a 3-D space with 0 [V] at the ground. The induced surface charge on the conductors is calculated. When applying BEM to the above electrostatic field analysis, we divide the surface of the conductor into triangular elements and use step functions as the basis function for BEM. By using the BEM analysis, we provide the BLR-matrices with $N = 1{,}000, 3{,}042, 4{,}950, 6{,}936, 10{,}400, 20{,}000, 50{,}864, 69{,}312, 101{,}250, 211{,}750$, and $338{,}000$, while varying the required accuracy for ACA as $\varepsilon = 1.0\text{e}-4, 1.0\text{e}-6$, and $1.0\text{e}-8$. The block size l of BLR is set to be $l = 10\sqrt{N}$. **Table 3** shows the observed memory usage, the maximum rank and the average rank for the BLR-matrices of $N = 101{,}250, 211{,}750$, and $338{,}000$. **Figure 7** shows the distribution of rank of low-rank submatrices in the first row for the BLR-matrices with $N = 101{,}250$ and $211{,}750$.

Table 3 Specification of BLR-matrices for the static electric field problem with the sphere model of N when the required accuracy is set to be ε.

Matrix size N	Accuracy ε	Memory [MB]	Maximum Rank	Average Rank
101,250	1.0e-4	3,244	191	18
	1.0e-6	3,947	279	29
	1.0e-8	4,754	371	43
211,750	1.0e-4	9,141	173	15
	1.0e-6	10,960	291	25
	1.0e-8	12,953	372	36
338,000	1.0e-4	14,029	199	10
	1.0e-6	17,469	274	17
	1.0e-8	21,509	350	24

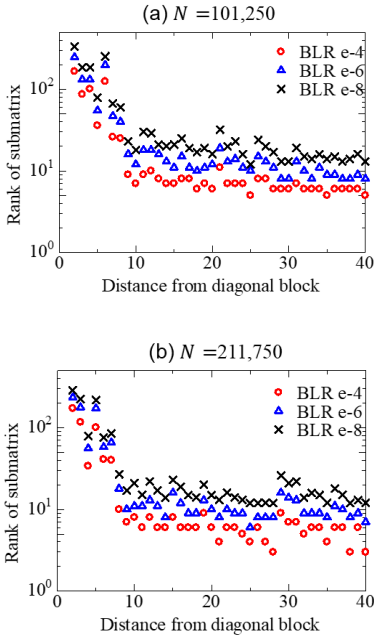


Fig. 7 Distribution of rank of low-rank submatrices. The submatrices next to the diagonal block have the maximum rank. The rank of submatrices decreases rapidly based on the distance from the diagonal block up to 10 block distance, and the rank of far off-diagonal submatrices is almost constant.

5.3 Quality of the Approximated QR Factorization

As mentioned in Section 4.2, our proposed MBGS algorithm

provides the approximation of QR factorization of the BLR-matrix: $\tilde{A} \approx \tilde{Q}\tilde{R}$, where $\tilde{Q}$ is the approximated orthogonal matrix and $\tilde{R}$ denotes the approximated upper triangular matrix. In addition to the error $\|\tilde{A}-\tilde{Q}\tilde{R}\|/\|\tilde{A}\|$, the orthogonality of the resulting matrix $\tilde{Q}$ is an important index to evaluate the quality of the QR factorization. Even in the case of a non-approximated MBGS algorithm for dense matrices, the error and orthogonality could decay as the matrix size increases because of round-off errors. We investigate the accuracy of the error and the orthogonality of $\tilde{Q}$ derived from our algorithm for BLR-matrices that were introduced in Section 5.2 and observe dependency on matrix size.

Instead of $\|\tilde{A} - \tilde{Q}\tilde{R}\|/\|\tilde{A}\|$, we calculate $\|\tilde{A}x - \tilde{Q}\tilde{R}x\|/\|\tilde{A}x\|$ with the vector x whose components are equal to one, because the direct multiplication $\tilde{Q}\tilde{R}$ requires the approximation in Fig. 4 which comes from the low-rank summation. Although we could directly evaluate $\|\tilde{A}-\tilde{Q}\tilde{R}\|/\|\tilde{A}\|$ if we treat the low-rank matrices $\tilde{A}$, $\tilde{Q}$, $\tilde{R}$ as dense matrices, the computational cost of $O(N^3)$ and the memory usage of $O(N^2)$ are required. The results are plotted as a function of matrix size N in **Fig. 8** (a). The error is the same order as the accuracy ε of the BLR-matrix $\tilde{A}$ to be factorized.

When using the BLR-matrix format in Eq. (3), it is inefficient to investigate the orthogonality vector-by-vector because each vector in the column direction is not explicitly stored. Therefore, we calculate the matrices $\tilde{Q}^T_{*,i}\tilde{Q}_{*,j}$ for $1 \le i, j \le N_b$, which are expected to satisfy the following relation:

$$\tilde{Q}^T_{*,i}\tilde{Q}_{*,j} = \begin{cases} I & (i = j) \\ 0 & (i \neq j). \end{cases} \tag{11}$$

Since the TSQR($\tilde{A}_{*,j}$) described in Section 4.2 is performed without any approximation, the error in $\|\tilde{Q}^T_{*,i}\tilde{Q}_{*,j} - I\|_2$ must be smaller than the error in $\|\tilde{Q}^T_{*,i}\tilde{Q}_{*,j}\|_2$ $(i \neq j)$. We calculate the maximum value of $\|\tilde{Q}^T_{*,i}\tilde{Q}_{*,j}\|_2$ for all the test BLR-matrices, and plot the results as a function of matrix size N in Fig. 8 (b). For matrix sizes up to tens of thousands, the accuracy of orthogonality is the same order as the accuracy ε of the BLR-matrix $\tilde{A}$ to be factorized. Although the orthogonality decays as the matrix size increases, the accuracy is relatively preserved in the case where $\varepsilon = 1.0\mathrm{e}-4$.

5.4 The Computational Complexity of QR Factorization Using MBGS Algorithm for BLR-matrices

Here, we investigate the dependency of the matrix size N on the computational time of the QR factorization. We perform the factorization using our proposed method in Section 4.2 for the BLR-matrices introduced in Section 5.2. Moreover, we calculate the QR factorization of dense matrices using the Intel MKL library. In **Fig. 9**, the observed computational times are plotted as a function of the matrix size N. We can see from this figure that the computational complexity of QR factorization of dense matrices is $O(N^3)$ in accordance with the theory discussed in Section 3. However, the computational complexity of our proposed algorithm is clearly less than $O(N^3)$ and looks less than the theoretical $O(N^2)$estimated in Section 4.2. One possibility for this outcome is that the average ranks of BLR-matrices in Table 3 decrease as the matrix size N increases.

As expected, the calculation time of our proposed algorithm

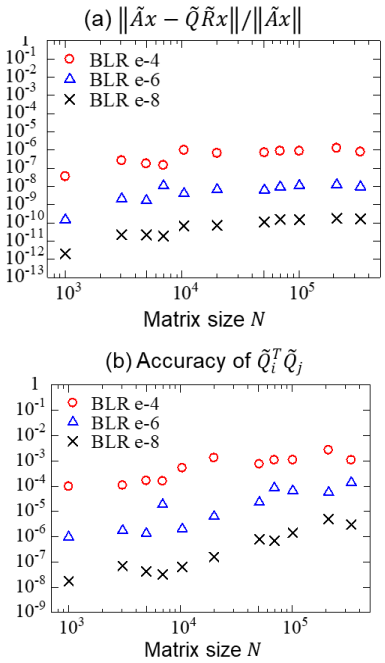


Fig. 8 The error of factorization (a) and accuracy of orthogonality of the resulting orthogonal matrix $\tilde{Q}$ (b) derived from our proposed MBGS algorithm for BLR-matrices.

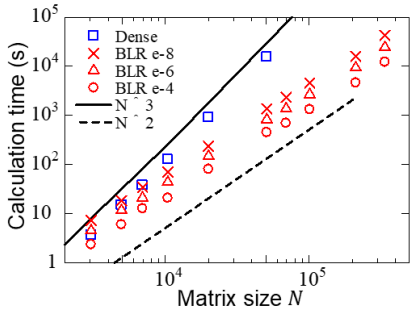


Fig. 9 Calculation time of the proposed algorithm and MKL routines ("dgeqrf" and "dorgqr").

depends on the accuracy of BLR-matrices. The cases where $\varepsilon = 1.0\mathrm{e}-4$ are about 3.5 fold faster than the cases where $\varepsilon = 1.0\mathrm{e}-8$ regardless of the matrix size N. For small matrices, the implementation of our proposed algorithm is slower than the MKL library when high accuracy is required. However, for the cases where $\varepsilon = 1.0\mathrm{e}-4$, our implementation is significantly faster than the MKL library regardless of the matrix size N and achieves the QR factorization of a matrix of $N = 338{,}000$ on a single node with 32 GB memory.

5.5 Performance of Parallel MBGS for BLR-matrices

First, we confirm the parallel scalability using OpenMP threads in a CPU socket. For the test matrices with $N = 101{,}205$ and 211,750 in Table 3, we observe the computational time of our proposed parallel MBGS algorithm in Section 4.3 when varying the number of OpenMP threads from 1 to 10 in a single MPI process. **Table 4** shows the results. We observe a parallel speed-up up to 10 threads, and the calculation time with 10 threads is about 5-fold faster than the case with a single thread. The bottleneck for the parallel speed-up is the calculation for $\tilde{Q}^T_{*j}\tilde{A}_{*k}$ in line 9 in Fig. 5.

Table 4 Parallel scalability using OpenMP threads in a CPU socket when performing parallel MBGS for BLR-matrices with a weak admissibility condition where $\varepsilon = 1.0\mathrm{e}-8$.

(a) BLR-matrix with N=211,750

#Threads	Time_all (s)	TSQR	$\tilde{Q}_{*j}^T \tilde{A}_{*k}$	$\tilde{A}_{*k} - \tilde{Q}_{*j}\,\tilde{R}_{jk}$
1	15,800	999	7,480	7,321
2	8,801	512	4,656	3,633
4	5,153	282	3,029	1,842
5	4,449	238	2,708	1,503
6	3,996	203	2,503	1,290
8	3,537	165	2,338	1,034
10	3,202	142	2,143	917

(b) BLR-matrix with N=101,250

#Threads	Time_all	TSQR	$\tilde{Q}_{*j}^T \tilde{A}_{*k}$	$\tilde{A}_{*k} - \tilde{Q}_{*j}\,\tilde{R}_{jk}$
1	4,515	272	2,149	2,094
2	2,594	146	1,399	1,049
4	1,548	80	929	539
5	1,345	70	836	439
6	1,225	61	793	371
8	1,055	50	716	289
10	949	44	661	244

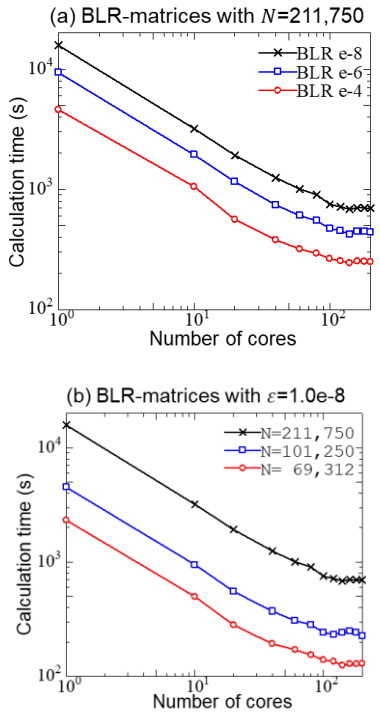


Fig. 10 Parallel scalability using the hybrid MPI+OpenMP on a distributed memory system when performing parallel MBGS for BLR-matrices.

Next, we investigate the parallel scalability using the hybrid MPI+OpenMP on the distributed memory system and observe the computational time when varying the number of cores from 1 to 200. In the calculations, we use a single MPI process with 10 OpenMP threads per socket, which means two MPI processes per a node. **Figure 10** (a) shows the results. Parallel speed-up up to about 140 cores is observed. The fastest time is about 20-fold faster than the time of serial computation. As Fig. 10 (b) shows, similar results are observed if we use test matrices of different sizes.

6. Conclusions

In this paper, we implemented a QR factorization based on the BLR matrix format with a weak admissibility condition. We used a modified block Gram Schmidt (MBGS) algorithm for BLR-matrices with two constraints on the structure and rank of the resulting matrices. We confirmed that the application requires only arithmetic operations related to low-rank submatrices in addition to the usual ones of dense submatrices, thanks to the constrained lattice structure of BLR-matrices, which is similar to the format of a block divided dense matrix. Although the QR factorization of dense square matrices requires memory usage of $O(N^2)$ and computational costs of $O(N^3)$, they can be reduced to $O(N^{1.5})$ and $O(N^2)$, respectively, when our proposed MBGS is employed to BLR-matrices with a block size $l \propto \sqrt{N}$. We confirmed this both in theory and in experiments. Thanks to the reduced complexity, we achieved the approximated QR factorization of a matrix with $N = 338{,}000$ on a single node with 32 GB memory. As future work, we will consider the use of more efficient low-rank structured matrices with a similar lattice structure, such as lattice $\mathcal{H}$-matrices [12] and multilevel BLR [24]. It is expected in theory that a QR factorization using the MBGS algorithm based on lattice $\mathcal{H}$-matrices requires only memory usage of $O(N \log N)$ and computational costs of $O(N \log^2 N)$.

Our proposed QR factorization based on BLR-matrices provides the approximation of the orthogonal matrix $\tilde{Q}$ and the upper triangular matrix $\tilde{R}$. In the numerical experiments, we confirmed that the factorization error and the accuracy of the orthogonality depends on the accuracy of the BLR-matrix to be factorized. This means that the accuracy of the QR factorization is controllable. Since the calculation time also depends on the accuracy of BLR-matrices, we may deliberately choose low accuracy and use it as a preconditioner for Krylov subspace methods. For large matrices, we observed a decay of the orthogonality when we require high accuracy. To avoid this problem, the use of quadruple-precision and/or increasing the rank of the resulting matrices $\tilde{Q}$ and $\tilde{R}$ may be considered. Instead of using MBGS, another promising approach would employ the tile QR factorization based on Householder reflectors [25], if the reflectors could be efficiently approximated by the low-rank structured matrices.

We also proposed the parallelization algorithm of the MBGS for BLR-matrices, and implemented the algorithm using the hybrid MPI+OpenMP programming model. Making full use of the simple structure of BLR-matrices and a block-cyclic assignment strategy to balance the load, the implementation achieves parallel scalability up to at least 140 cores. The fastest time using 140 cores is about 20-fold faster than the time of serial computation. For further improvement in the near future, we will introduce a 2D cyclic assignment strategy to parallelize the operations within a block column.

Acknowledgments This work was partially supported by JSPS KAKENHI Grant Numbers 17K19962, 17H01749, 18H03248, 19H04122 and "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" and "High Performance Computing Infrastructure" in Japan (Project ID: jh190043). Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy National Nuclear Security Administration

under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] Hackbusch, W.: A sparse matrix arithmetic based on-matrices, Part I: Introduction to H-matrices, *Computing*, Vol.62, No.2, pp.89–108 (1999).

[2] Bebendorf, M. and Hackbusch, W.: Existence of H-matrix approximants to the inverse FE-matrix of elliptic operators with L8-coefficients, *Numer. Math.*, Vol.95, No.1, pp.1–28 (2003).

[3] Amestoy, P., Ashcraft, C., Boiteau, O., Buttari, A., L'Excellent, J.-Y. and Weisbecker, C.: Improving multifrontal methods by means of block low-rank representations, *SIAM Journal on Scientific Computing*, Vol.37, No.3, pp.A1451–A1474 (2015).

[4] Ida, A., Nakashima, H. and Kawai, M.: Parallel Hierarchical Matrices with Block Low-rank Representation on Distributed Memory Computer Systems, *Proc. International Conference on High Performance Computing in Asia-Pacific Region*, pp.232–240, ACM (2018).

[5] Wolfgang, H. and K. Boris, N. and Ronald, K.: Hierarchical matrices based on a weak admissibility criterion: *Computing*, Vol.73, No.3, pp.207–243 (2004).

[6] Akbudak, K., Ltaief, H., Mikhalev, A. and Keyes, D.: Tile low rank Cholesky factorization for climate/weather modeling applications on manycore architectures, *International Supercomputing Conference*, pp.22–40 (2017).

[7] Benner, P., Mach, T. and Seaside, C.A.: On the QR decomposition of H-matrices, *Computing*, Vol.88, No.3–4, pp.111–129 (2010).

[8] Bebendorf, M. and Kriemann, R.: Fast parallel solution of boundary integral equations and related problems, *Computing and Visualization in Science*, Vol.8, No.3, pp.121–135 (2005).

[9] Ida, A., Iwashita, T., Mifune, T. and Takahashi, Y.: Parallel hierarchical matrices with adaptive cross approximation on symmetric multiprocessing clusters, *Journal of Information Processing*, Vol.22, No.4, pp.642–650 (2014).

[10] Chandrasekaran, S., Dewilde, P., Gu, M., Lyons, W. and Pals, T.: A fast solver for HSS representations via sparse matrices, *SIAM J. Matrix Anal. Appl.*, Vol.29, No.1, pp.67–81 (2006).

[11] Ambikasaran, S. and Darve, E.: An O(N log N) Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices, *Journal of Scientific Computing*, Vol.57, No.3, pp.477–501 (2013).

[12] Ida, A.: Lattice H-matrices on distributed-memory systems, *2018 IEEE International Parallel and Distributed Processing Symposium* (*IPDPS*), pp.389–398, IEEE (2018).

[13] Bouwmeester, H., Jacquelin, M., Langou, J. and Robert, Y.: Tiled QR factorization algorithms, *Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 7, ACM (2011).

[14] Bosilca, G., Bouteiller, A., Danalis, A., Herault, T., Lemarinier, P. and Dongarra, J.: DAGuE: A generic distributed DAG engine for high performance computing, *Parallel Comput.*, Vol.38, No.1–2, pp.37–51 (2012).

[15] Yamazaki, I., Tomov, S. and Dongarra, J.: Mixed-precision Cholesky QR factorization and its case studies on multicore CPU with multiple GPUs, *SIAM Journal on Scientific Computing*, Vol.37, No.3, pp.C307–C330 (2015).

[16] Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. and Whaley, R.C.: ScaLAPACK Users' Guide, *Society for Industrial and Applied Mathematics*, Philadelphia, PA (1997).

[17] Demmel, J., Grigori, L., Hoemmen, M. and Langou, J.: Communication-optimal parallel and sequential QR and LU factorizations, *SIAM Journal on Scientific Computing*, Vol.34, No.1, pp.A206–A239 (2012).

[18] Jalby, W. and Philippe, B.: Stability analysis and improvement of the block Gram–Schmidt algorithm, *SIAM Journal on Scientific and Statistical Computing*, Vol.12, No.5, pp.1058–1073 (1991).

[19] Ida, A., Iwashita, T., Ohtani, M. and Hirahara, K.: Improvement of hierarchical matrices with adaptive cross approximation for largescale simulation, *Journal of Information Processing*, Vol.23, No.3, pp.366–372 (2015).

[20] Bebendorf, M.: Hierarchical matrices, pp.15–16, Springer (2008).

[21] "HACApK", available from ⟨https://github.com/Post-Peta-Crest/ppOpenHPC/tree/MATH/HACApK⟩ (accessed 2019-04-04).

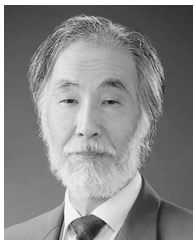
[22] Bebendorf, M.: Approximation of boundary element matrices, *Numer. Math.*, Vol.86, No.4, pp.565–589 (2000).

[23] Iwashita, T., Ida, A., Mifune, T. and Takahashi, Y.: Software Framework for Parallel BEM Analyses with H-matrices Using MPI and OpenMP, *Procedia Computer Science*, Vol.108, pp.2200–2209 (2017).

[24] Amestoy, P., Buttari, A., L'Excellent, J.-Y. and Mary, T.: Bridging the gap between flat and hierarchical low-rank matrix formats: The multilevel BLR format, The University of Manchester MIMS EPrints (2018).

[25] Bouwmeester, H., Jacquelin, M., Langou, J. and Robert, Y.: Tiled QR factorization algorithms, *Proc. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, No.7, ACM (2011).

Akihiro Ida was born in Japan in 1971. He received B.Math and M.E. degrees from Nagoya University in 1994 and 1996, respectively. In 2008, Chuo University awarded him a Ph.D. degree in mathematics. In 2000–2012, he researched and developed linear solvers at VINAS Co., Ltd. In 2012–2015, he worked as an assistant professor in the Academic Center for Computing and Media Studies, Kyoto University. He currently works as an associate professor in the Information Technology Center, The University of Tokyo. His research interests include discretization methods for integro-differential equations, numerical linear algebra and high performance computing.



Hiroshi Nakashima received his M.E. and Ph.D. from Kyoto University in 1981 and 1991 respectively, and was engaged in research on inference systems with Mitsubishi Electric Corporation from 1981. He became an associate professor at Kyoto University in 1992, a professor at Toyohashi University of Technology in 1997, and a professor at Kyoto University in 2006. His current research interests are in high-performance computing systems and programming on them. He received the Motooka award in 1988 and the Sakai award in 1993. He is a Fellow of IPSJ, and a member of IEEE-CS, ACM, ALP and TUG.

Tasuku Hiraishi was born in 1981. He received his B.E. in Information Science in 2003, an Master of informatics in 2005, and his Ph.D. in informatics in 2008, all from Kyoto University. In 2007–2008, he was a fellow of the JSPS (at Kyoto University). Since 2008, he has been working at Kyoto University as an assistant professor at Supercomputing Research Laboratory, Academic Center for Computing and Media Studies, Kyoto University. His research interests include parallel programming languages and high performance computing. He won the IPSJ Best Paper Award in 2010. He is a member of IPSJ, JSSST, and ACM.

Ichitaro Yamazaki received his Ph.D. degree in Computer Science from the University of California at Davis in 2008. He is currently a research scientist at the Sandia National Laboratories, where his interests lie in high-performance computing, especially for linear algebra and scientific computing. Before joining the Sandia National Laboratories, he has also worked in the Innovative Computing Laboratory at the University of Tennessee at Knoxville as a research scientist from 2011 to 2019, and in Scientific Computing Group at Lawrence Berkeley National Laboratory from 2008 to 2011, as a Postdoctoral Researcher.

Rio Yokota is an Associate Professor at GSIC, Tokyo Institute of Technology. He was a Research Scientist at ECRC, KAUST from September 2011 to March 2015 before joining Tokyo Tech. His research interests range from high performance computing, hierarchical low-rank approxima-tion methods, and scalable deep learning. He was part of the team that won the Gordon Bell prize for price/performance in 2009. He is a member of ACM, IEEE, SIAM, IPSJ, JSAI, and JSIAM.

Takeshi Iwashita was born in 1971. He received a B.E., an M.E., and a Ph.D. from Kyoto University in 1992, 1995, and 1998, respectively. In 1998–1999, he worked as a post-doctoral fellow of the JSPS project in the Graduate School of Engineering, Kyoto University. He moved to the Data Processing Center of the same university in 2000. In 2003–2014, he worked as an associate professor in the Academic Center for Computing and Media Studies, Kyoto University. He currently works as a professor in the Information Initiative Center, Hokkaido University. His research interests include high performance computing, linear iterative solver, and electromagnetic field analysis. He is a member of IEEE, SIAM, IPSJ, IEEJ, JSIAM, JSCES, and JSAEM.