Comput Mech (2017) 60:379–392
DOI 10.1007/s00466-017-1407-2

ORIGINAL PAPER

A new fast direct solver for the boundary element method

S. Huang[1] · **Y. J. Liu**[1,2]

Received: 16 November 2016 / Accepted: 30 March 2017 / Published online: 18 April 2017

Abstract A new fast direct linear equation solver for the boundary element method (BEM) is presented in this paper. The idea of the new fast direct solver stems from the concept of the hierarchical off-diagonal low-rank matrix. The hierarchical off-diagonal low-rank matrix can be decomposed into the multiplication of several diagonal block matrices. The inverse of the hierarchical off-diagonal low-rank matrix can be calculated efficiently with the Sherman–Morrison–Woodbury formula. In this paper, a more general and efficient approach to approximate the coefficient matrix of the BEM with the hierarchical off-diagonal low-rank matrix is proposed. Compared to the current fast direct solver based on the hierarchical off-diagonal low-rank matrix, the proposed method is suitable for solving general 3-D boundary element models. Several numerical examples of 3-D potential problems with the total number of unknowns up to above 200,000 are presented. The results show that the new fast direct solver can be applied to solve large 3-D BEM models accurately and with better efficiency compared with the conventional BEM.

Keywords Fast direct solver · Boundary element method · 3-D potential problem

✉ Y. J. Liu
Yijun.Liu@uc.edu

[1] Mechanical Engineering, University of Cincinnati, P. O. Box 210072, Cincinnati, OH 45221-0072, USA

[2] Institute for Computational Mechanics and its Applications, Northwestern Polytechnical University, Xi'an 710072, Shaanxi, China

1 Introduction

The boundary element method (BEM) [1–6] has been used to solve different types of problems for decades. One advantage of the BEM is that only the boundary of the domain is discretized, which reduces the number of elements substantially. However, the efficiency in solution has been the most challenging problem for the BEM in analyzing large-scale problems. The reason is that the conventional BEM needs long CPU time to calculate the coefficient matrix and solve the linear system of equations. It also needs large computer memory to store the coefficient matrix. The conventional BEM generates a dense and unsymmetrical coefficient matrix. Calculating the matrix requires $O(N^2)$ operations and solving the linear system of equations requires $O(N^3)$ operations with a traditional direct solver (with N being the number of equations in the linear system). As N becomes larger, the required CPU time and computer memory increase quickly.

In the mid of 1980s, Rokhlin and Greengard [7,8] proposed the fast multipole method (FMM) and it has been applied to accelerate the solutions of the BEM equations effectively. The basic idea behind the fast multipole BEM is to translate the element-to-element interaction to the cell-to-cell interaction. To implement the translation, a tree structure is usually built to divide all the elements into some cells and the multipole expansions of the kernel functions are derived. An iterative solver is usually used for the fast multipole BEM, where the matrix-vector multiplication is performed by calculating the fast multipole expansions and translations. Thanks to the FMM, the computational complexity of the BEM can be reduced to near $O(N)$. The fast multipole BEM has been further developed for solving potential [9–11], elasticity [12–14], and acoustic problems [15–17] by the authors and their colleagues, as well as for solving electromagnet-

ics problems [18–21] by others. Comprehensive reviews and detailed discussions of the fast multipole BEM can be found in Refs. [6,22–24].

The adaptive cross approximation (ACA) has also been used to improve the computational efficiency of the BEM. In 1990s, Hackbusch et al. [25–28] introduced the concept of the hierarchical matrix (H-matrix). By considering a dense matrix and dividing it hierarchically into submatrices, Hackbusch et al. indicated that some of the submatrices can be well- approximated by low-rank matrices. This type of hierarchical matrices occurs especially in the context of boundary integral equations (BIEs) [29]. Based on the theory of the hierarchical matrix, Bebendorf et al. developed the ACA method and applied it to the BEM [30,31]. Since the ACA was fully developed from the algebra of the matrix, it is not necessary to derive the analytical expansions of kernel functions. Therefore, the ACA BEM has also become popular in many applications [32–35].

The success of the fast multipole BEM and ACA BEM relies on the development of the iterative solver. Saad [36] proposed the GMRES solver, which can be applied to solve a linear system of equations without knowing the matrix explicitly. With the help of GMRES, the fast multipole BEM and ACA BEM only need to be applied to perform the matrix-vector multiplication in each iteration. However, the speed of convergence is always a concern when using an iterative solver. The number of iterations to obtain the solution depends highly on the condition number of the coefficient matrix. The smaller the condition number is, the faster the iterative solver converges. A preconditioner is frequently used to speed up the convergence. A common choice of the pre-conditioner for the fast multipole BEM or ACA BEM is the block-diagonal preconditioner because it is cheap to calculate and store. Many other efficient preconditioners are also available [37]. Another drawback of using iterative solvers is that they cannot handle multiple right-hand side vectors for a linear system of equations. Therefore, the system needs to be solved multiple times for all the given right-hand side vectors (such as load cases).

To avoid the convergence problem with iterative solvers, many research groups started to develop fast direct solvers for the dense matrix. In recent years, Martinsson's group [38,39], Greengard's group [40–42], and Darve's group [43,44] proposed various fast direct solvers by utilizing the theory of H-matrix. The common idea behind these algorithms is to divide the matrix hierarchically, construct the low-rank approximation for certain submatrices and perform a fast update to the solution recursively. One type of the hierarchical matrix is called hierarchical off-diagonal low-rank matrix (or HODLR matrix). The most important feature of this type of matrix is that all the off-diagonal submatrices can be approximated by low-rank matrices. Because of this feature, a HODLR matrix can be decomposed into the multiplication of several diagonal block matrices. The inverse of the HODLR matrix can be calculated efficiently with the Sherman–Morrison–Woodbury formula [45,46].

To the authors' best knowledge, all the current fast direct solvers based on the HODLR matrix have been tested for solving 1-D and 2-D problems. No general 3-D BEM models using the fast direct solver based on the HODLR matrix have been reported in the literature. In this paper, we present a new fast direct solver based on the HODLR matrix for solving general 3-D BEM models. The paper is organized as follows: In Sect. 2, the BIE for potential problems is reviewed. In Sect. 3, the hierarchical off-diagonal low-rank matrix is introduced. In Sect. 4, the formulation of the new fast direct solver is proposed. In Sect. 5, numerical examples are presented to show the accuracy and efficiency of the new fast direct solver. In Sect. 6, conclusions and discussions on the future improvements of the new fast direct solver are provided.

2 Boundary integral equations for potential problems

Consider a potential problem in a 2-D or 3-D domain V with the boundary S. The governing equation for a potential problem is:

$$\nabla^2 \phi\left(\mathbf{x}\right) = 0, \quad \forall \mathbf{x} \in V, \tag{1}$$

and the boundary conditions are:

$$\begin{aligned} \phi\left(\mathbf{x}\right) &= \bar{\phi}\left(\mathbf{x}\right), \quad \forall \mathbf{x} \in S_\phi, \\ q\left(\mathbf{x}\right) &= \bar{q}\left(\mathbf{x}\right), \quad \forall \mathbf{x} \in S_q, \end{aligned} \tag{2}$$

where $\phi\left(\mathbf{x}\right)$ is the potential and $q\left(\mathbf{x}\right)$ is the normal derivative of $\phi\left(\mathbf{x}\right)$; $\bar{\phi}\left(\mathbf{x}\right)$ and $\bar{q}\left(\mathbf{x}\right)$ are given functions on S_ϕ and S_q, respectively; $S = S_\phi \cup S_q$ and $S_\phi \cap S_q = \varnothing$.

With the help of Green's second identity, the conventional boundary integral equation (CBIE) for the potential problem is [6]:

$$\begin{aligned} c\left(\mathbf{x}\right)\phi\left(\mathbf{x}\right) = &\int_S G\left(\mathbf{x},\mathbf{y}\right) q\left(\mathbf{y}\right) dS \\ &- \int_S F\left(\mathbf{x},\mathbf{y}\right)\phi\left(\mathbf{y}\right) dS, \quad \mathbf{x} \in S, \end{aligned} \tag{3}$$

where $G\left(\mathbf{x},\mathbf{y}\right)$ is the fundamental solution, with $\mathbf{x}$ being the source (collocation) point and $\mathbf{y}$ the field (integration) point. The value of the coefficient $c\left(\mathbf{x}\right)$ depends on the smoothness of the curve (2-D) or surface (3-D) near point $\mathbf{x}$. When the boundary is smooth at $\mathbf{x}$, $c\left(\mathbf{x}\right) = 1/2$. For 3-D potential problems, we have:

$$G(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi r}, \quad r = |\mathbf{x}-\mathbf{y}|. \tag{4}$$

$F(\mathbf{x},\mathbf{y})$ is the normal derivative of $G(\mathbf{x},\mathbf{y})$ at the field point $\mathbf{y}$.

To solve the BIE, the boundary S is discretized into N boundary elements. The final linear system of equations can be written as:

$$\mathbf{A}\boldsymbol{\lambda} = \mathbf{b} \tag{5}$$

where $\mathbf{A}$ is the coefficient matrix, $\boldsymbol{\lambda}$ is the vector of unknowns on the boundary, and $\mathbf{b}$ is the right-hand side vector.

3 Hierarchical off-diagonal low-rank matrix

In Sect. 1, we mentioned that most of the current fast direct solvers are based on the theory of H-matrix. In Ref. [29], Ambikasaran presented the classification of H-matrix and the relationships between different types of H-matrix. One type of H-matrix is called the hierarchically off-diagonal low-rank (HODLR) matrix. If a dense matrix is hierarchically divided and all the off-diagonal submatrices can be well-approximated by low-rank matrices, we call it as a HODLR matrix. In this section, the definition of the HODLR matrix is reviewed.

3.1 Definition of HODLR matrix

The HODLR matrix is a type of matrix that the off-diagonal submatrices can be approximated by low-rank matrices. Here are examples of the HODLR matrix:

$$\mathbf{K} \approx \mathbf{K}^1 = \begin{bmatrix} \mathbf{K}_1^1 & \mathbf{U}_1^1\left(\mathbf{V}_1^1\right)^T \\ \mathbf{U}_2^1\left(\mathbf{V}_2^1\right)^T & \mathbf{K}_2^1 \end{bmatrix}$$

$$\mathbf{K} \approx \mathbf{K}^2 = \begin{bmatrix} \begin{bmatrix} \mathbf{K}_1^2 & \mathbf{U}_1^2\left(\mathbf{V}_1^2\right)^T \\ \mathbf{U}_2^2\left(\mathbf{V}_2^2\right)^T & \mathbf{K}_2^2 \end{bmatrix} & \mathbf{U}_1^1\left(\mathbf{V}_1^1\right)^T \\ \mathbf{U}_2^1\left(\mathbf{V}_2^1\right)^T & \begin{bmatrix} \mathbf{K}_3^2 & \mathbf{U}_3^2\left(\mathbf{V}_3^2\right)^T \\ \mathbf{U}_4^2\left(\mathbf{V}_4^2\right)^T & \mathbf{K}_4^2 \end{bmatrix} \end{bmatrix}. \tag{6}$$

In the first equation of Eq. (6), $\mathbf{K}$ is approximated by $\mathbf{K}^1$. $\mathbf{K}^1$ is divided into four submatrices. $\mathbf{K}_1^1$ and $\mathbf{K}_2^1$ are two diagonal submatrices. Two off-diagonal matrices are approximated by the multiplication of two matrices $\mathbf{U}_i^1$ and $\mathbf{V}_i^1$ $(i = 1, 2)$, respectively. $\mathbf{U}_i^1$ and $\mathbf{V}_i^1$ $(i = 1, 2) \in \mathbb{R}^{N_i \times p_i}$ where $p_i \ll N_i$. $\mathbf{K}^1$ is called as a 1-level HODLR matrix. In the second equation of Eq. (6), $\mathbf{K}_1^1$ and $\mathbf{K}_2^1$ are further divided into four submatrices, respectively. After two divisions, $\mathbf{K}$ is approximated by $\mathbf{K}^2$, which is called as a 2-level HODLR matrix. In this paper, the superscripts of a matrix denote the level of the matrix. In general, $\mathbf{K}^l$ is called l-level HODLR matrix. The i th diagonal submatrix of l-level HODLR matrix can be written as:

$$\mathbf{K}_i^l = \begin{bmatrix} \mathbf{K}_{2i-1}^l & \mathbf{U}_{2i-1}^l\left(\mathbf{V}_{2i-1}^l\right)^T \\ \mathbf{U}_{2i}^l\left(\mathbf{V}_{2i}^l\right)^T & \mathbf{K}_{2i}^l \end{bmatrix} \tag{7}$$

where $i = 1, 2, 3, \ldots, 2^{l-1}$. $\mathbf{K}_{2i}^l$ and $\mathbf{K}_{2i-1}^l$ are square matrices. Their dimensions are N_{2i}^l and N_{2i-1}^l, respectively. $\mathbf{U}_{2i}^l$ and $\mathbf{V}_{2i}^l$ are $N_{2i}^l \times p_{2i}^l$ matrices. $\mathbf{U}_{2i-1}^l$ and $\mathbf{V}_{2i-1}^l$ are $N_{2i-1}^l \times p_{2i-1}^l$ matrices. Usually, we have $p_{2i-1}^l \ll N_{2i-1}^l$ and $p_{2i}^l \ll N_{2i}^l$. It is worth to mention that N_{2i}^l and N_{2i-1}^l are not necessarily equal, so are p_{2i}^l and p_{2i-1}^l. This definition is different from the definition in Ref. [43] where the dimensions of $\mathbf{U}_{2i}^l$, $\mathbf{V}_{2i}^l$, $\mathbf{U}_{2i-1}^l$ and $\mathbf{V}_{2i-1}^l$ are all $N/2^l \times p$. This modification does not affect the following matrix factorization and makes the definition of HODLR matrix more general and flexible, especially when N is not the power of 2.

3.2 HODLR matrix factorization

In this section, we discuss the factorization of HODLR matrix. The overall idea is to factor the dense matrix $\mathbf{K}$ into the multiplication of several diagonal-block matrices $\mathbf{K}_l$ $(l = 1, \ldots, l_\mathrm{m})$. The HODLR matrix at level l_m can be written as:

$$\mathbf{K}^{l_\mathrm{m}} = \begin{bmatrix} \begin{bmatrix} \mathbf{K}_1^{l_\mathrm{m}} & \mathbf{U}_1^{l_\mathrm{m}}\left(\mathbf{V}_1^{l_\mathrm{m}}\right)^T \\ \mathbf{U}_2^{l_\mathrm{m}}\left(\mathbf{V}_2^{l_\mathrm{m}}\right)^T & \mathbf{K}_2^{l_\mathrm{m}} \end{bmatrix} & \mathbf{U}_1^{l_\mathrm{m}-1}\left(\mathbf{V}_1^{l_\mathrm{m}-1}\right)^T & \cdots & \cdots \\ \mathbf{U}_2^{l_\mathrm{m}-1}\left(\mathbf{V}_2^{l_\mathrm{m}-1}\right)^T & \begin{bmatrix} \mathbf{K}_3^{l_\mathrm{m}} & \mathbf{U}_3^{l_\mathrm{m}}\left(\mathbf{V}_3^{l_\mathrm{m}}\right)^T \\ \mathbf{U}_4^{l_\mathrm{m}}\left(\mathbf{V}_4^{l_\mathrm{m}}\right)^T & \mathbf{K}_4^{l_\mathrm{m}} \end{bmatrix} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots \\ \vdots & \vdots & \vdots & \begin{bmatrix} \mathbf{K}_{2^{l_\mathrm{m}}-1}^{l_\mathrm{m}} & \mathbf{U}_{2^{l_\mathrm{m}}-1}^{l_\mathrm{m}}\left(\mathbf{V}_{2^{l_\mathrm{m}}-1}^{l_\mathrm{m}}\right)^T \\ \mathbf{U}_{2^{l_\mathrm{m}}}^{l_\mathrm{m}}\left(\mathbf{V}_{2^{l_\mathrm{m}}}^{l_\mathrm{m}}\right)^T & \mathbf{K}_{2^{l_\mathrm{m}}}^{l_\mathrm{m}} \end{bmatrix} \end{bmatrix} \tag{8}$$

The first step of factorization is to factor the diagonal-block matrices at level l_{m}:

$$\mathbf{K}_{l_{\mathrm{m}}} = \begin{bmatrix} \mathbf{K}_1^{l_{\mathrm{m}}} & & \\ & \mathbf{K}_2^{l_{\mathrm{m}}} & \\ & & \mathbf{K}_{2^{l_{\mathrm{m}}}}^{l_{\mathrm{m}}} \end{bmatrix} \tag{9}$$

After $\mathbf{K}_{l_{\mathrm{m}}}$ is factorized, we get the HODLR matrix at level $l_{\mathrm{m}}-1$:

$$\mathbf{K}^{l_{\mathrm{m}}-1} = \begin{bmatrix} \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_1^{l_{\mathrm{m}}}\left(\mathbf{V}_1^{l_{\mathrm{m}}}\right)^T \\ \tilde{\mathbf{U}}_2^{l_{\mathrm{m}}}\left(\mathbf{V}_2^{l_{\mathrm{m}}}\right)^T & \mathbf{I} \end{bmatrix} & \tilde{\mathbf{U}}_1^{l_{\mathrm{m}}-1}\left(\mathbf{V}_1^{l_{\mathrm{m}}-1}\right)^T & \cdots & \cdots \\ \tilde{\mathbf{U}}_2^{l_{\mathrm{m}}-1}\left(\mathbf{V}_2^{l_{\mathrm{m}}-1}\right)^T & \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_3^{l_{\mathrm{m}}}\left(\mathbf{V}_3^{l_{\mathrm{m}}}\right)^T \\ \tilde{\mathbf{U}}_4^{l_{\mathrm{m}}}\left(\mathbf{V}_4^{l_{\mathrm{m}}}\right)^T & \mathbf{I} \end{bmatrix} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots \\ \vdots & \vdots & \vdots & \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_{2^{l_{\mathrm{m}}}-1}^{l_{\mathrm{m}}}\left(\mathbf{V}_{2^{l_{\mathrm{m}}}-1}^{l_{\mathrm{m}}}\right)^T \\ \tilde{\mathbf{U}}_{2^{l_{\mathrm{m}}}}^{l_{\mathrm{m}}}\left(\mathbf{V}_{2^{l_{\mathrm{m}}}}^{l_{\mathrm{m}}}\right)^T & \mathbf{I} \end{bmatrix} \end{bmatrix} \tag{10}$$

where $\tilde{\mathbf{U}}_i^l$ indicates the matrix $\mathbf{U}_i^l$ is updated after factoring out $\mathbf{K}_{l_{\mathrm{m}}}$. Define

$$\mathbf{K}_i^{l_m-1} = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_{2i-1}^{l_{\mathrm{m}}}\left(\mathbf{V}_{2i-1}^{l_{\mathrm{m}}}\right)^T \\ \tilde{\mathbf{U}}_{2i}^{l_{\mathrm{m}}}\left(\mathbf{V}_{2i}^{l_{\mathrm{m}}}\right)^T & \mathbf{I} \end{bmatrix} \quad i = 1, 2, 3, \ldots, 2^{l_m-1} \tag{11}$$

Equation (10) can be re-written as:

$$\mathbf{K}^{l_{\mathrm{m}}-1} = \begin{bmatrix} \mathbf{K}_1^{l_m-1} & \tilde{\mathbf{U}}_1^{l_{\mathrm{m}}-1}\left(\mathbf{V}_1^{l_{\mathrm{m}}-1}\right)^T & \cdots & \cdots \\ \tilde{\mathbf{U}}_2^{l_{\mathrm{m}}-1}\left(\mathbf{V}_2^{l_{\mathrm{m}}-1}\right)^T & \mathbf{K}_2^{l_m-1} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots \\ \vdots & \vdots & \vdots & \mathbf{K}_{2^{l_m-1}}^{l_m-1} \end{bmatrix} \tag{12}$$

Now, we can factor the diagonal-block matrices at level $l_{\mathrm{m}}-1$ and obtain the HODLR matrix at level $l_{\mathrm{m}}-2$:

$$\begin{aligned} \mathbf{K}^{l_{\mathrm{m}}-1} &= \mathbf{K}_{l_m-1}\mathbf{K}^{l_{\mathrm{m}}-2} \\ &= \begin{bmatrix} \mathbf{K}_1^{l_m-1} & & & \\ & \mathbf{K}_2^{l_m-1} & & \\ & & \ddots & \\ & & & \mathbf{K}_{2^{l_m-1}}^{l_m-1} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_1^{l_{\mathrm{m}}-1}\left(\mathbf{V}_1^{l_{\mathrm{m}}-1}\right)^T & \cdots & \cdots \\ \tilde{\mathbf{U}}_2^{l_{\mathrm{m}}-1}\left(\mathbf{V}_2^{l_{\mathrm{m}}-1}\right)^T & \mathbf{I} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \cdots \\ \vdots & \vdots & \vdots & \mathbf{I} \end{bmatrix} \end{aligned} \tag{13}$$

By keeping factorizing out the block-diagonal matrices at each level, the HODLR matrix at level l_{m} can be written as (Fig. 1):

$$\mathbf{K}^{l_{\mathrm{m}}} = \mathbf{K}_{l_m}\mathbf{K}_{l_m-1}\ldots\mathbf{K}_0 \tag{14}$$

Therefore, the solution of the equation $\mathbf{Kx} = \mathbf{b}$ is given as: $\mathbf{x} = \mathbf{K}_0^{-1}\ldots\mathbf{K}_{l_m-1}^{-1}\mathbf{K}_{l_m}^{-1}\mathbf{b}$. For $\mathbf{K}_{l_m}$, each diagonal block is a small matrix so that $\mathbf{K}_{l_m}^{-1}$ is easy to calculate. For $\mathbf{K}_l$ ($l = 0, 1, \ldots, l_m - 1$), the i th diagonal block has the following general form:

$$\begin{aligned} \mathbf{K}_i^l &= \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_{2i-1}^{l+1}\left(\mathbf{V}_{2i-1}^{l+1}\right)^T \\ \tilde{\mathbf{U}}_{2i}^{l+1}\left(\mathbf{V}_{2i}^{l+1}\right)^T & \mathbf{I} \end{bmatrix} \\ &= \mathbf{I} + \begin{bmatrix} \tilde{\mathbf{U}}_{2i-1}^{l+1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{U}}_{2i}^{l+1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \left(\mathbf{V}_{2i-1}^{l+1}\right)^T \\ \left(\mathbf{V}_{2i}^{l+1}\right)^T & \mathbf{0} \end{bmatrix} \\ &= \mathbf{I} + \mathbf{U}_i^l\left(\mathbf{V}_i^l\right)^T \end{aligned} \tag{15}$$

where

$$\mathbf{U}_i^l = \begin{bmatrix} \tilde{\mathbf{U}}_{2i-1}^{l+1} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{U}}_{2i}^{l+1} \end{bmatrix} \text{ and } \mathbf{V}_i^l = \begin{bmatrix} \mathbf{0} & \mathbf{V}_{2i}^{l+1} \\ \mathbf{V}_{2i-1}^{l+1} & \mathbf{0} \end{bmatrix}.$$

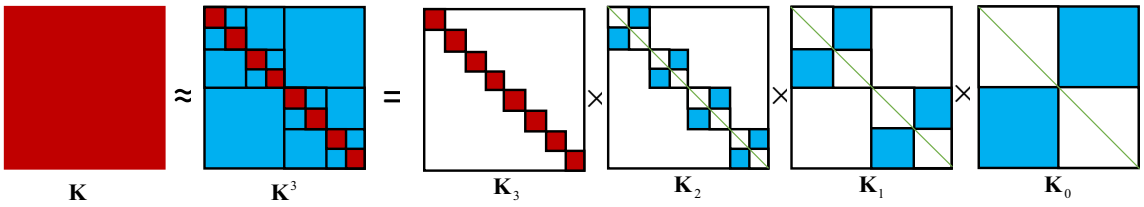

Fig. 1 HODLR matrix decomposition

To calculate the inverse matrix of $\mathbf{K}_l^l$, consider solving a system of the form:

$$\left(\mathbf{I}+\mathbf{U}\mathbf{V}^T\right)\mathbf{x}=\mathbf{b} \tag{16}$$

where $\mathbf{I}\in\mathbb{R}^{n\times n}$ is an identity matrix, $\mathbf{U},\mathbf{V}\in\mathbb{R}^{n\times p}$,$\mathbf{x},\mathbf{b}\in\mathbb{R}^{n\times r}$ and $p\le r<n$. The Sherman–Morrison–Woodbury formula [45,46] gives:

$$\mathbf{x}=\mathbf{b}-\mathbf{U}\left(\mathbf{I}+\mathbf{V}^T\mathbf{U}\right)^{-1}\mathbf{V}^T\mathbf{b} \tag{17}$$

The computational cost for solving (16) using (17) is $O\,(prn)$, which is much smaller than solving (16) by using LU decomposition directly.

4 Formulations of the new fast direct BEM solver

From the previous section, the HODLR matrix can be inverted easily because of the low-rank structure of all the off-diagonal submatrices. To develop the fast direct solver based on the HODLR matrix, we need an efficient approach to calculate the low-rank approximation of the off-diagonal submatrices.

In Ref. [42], Lai et al. developed a fast direct solver based on the HODLR matrix to solve the high-frequency electromagnetic scattering problems for 2-D large cavity. The ACA is used to calculate the low-rank approximation of all the off-diagonal submatrices in Lai's paper. However, in the traditional ACA BEM, the ACA is only used to approximate a certain type of the submatrices. In the common algorithm of ACA BEM, a tree structure is constructed by dividing all the elements into some clusters recursively. The clusters in the lowest level are usually called leaves of the tree structure. The division of the elements partitions the coefficient matrix. If two different clusters (a cluster pair) at the same level are well-separated geometrically, we consider that the corresponding off-diagonal submatrix can be approximated by low-rank matrices accurately and efficiently. To identify this kind of cluster pairs, an admissible condition is examined for every two clusters at the same level. If the admissible condition is satisfied, the corresponding coefficient submatrix is approximated by the ACA. Otherwise, this examining process is executed for the clusters at the lower levels. If two clusters at the lowest level of the tree structure still do not satisfy the admissible condition, the corresponding coefficient submatrix is calculated directly. Therefore, only the submatrices capturing the interaction of two admissible clusters are approximated by the ACA in the traditional ACA BEM. Generally, two different clusters in the higher levels of the tree structure are too close to satisfy the admissible condition. Directly using the ACA to approximate the corresponding submatrices will affect the efficiency and result in larger errors. Therefore, it is not reasonable to use the ACA to approximate all of the off-diagonal submatrices in a general case.

In this section, we propose a more general and efficient approach to approximate all the off-diagonal submatrices. With this approach, the ACA is only used to approximate the submatrices capturing the interaction of admissible cluster pairs. For the off-diagonal submatrices capturing the interaction of inadmissible cluster pairs, the randomized interpolative decomposition is applied to calculate their approximations. The resulting new fast direct solver is suitable for solving general 3-D BEM models.

4.1 BEM matrix approximation

For the new fast direct solver of BEM, a binary tree is also built to divide the boundary elements into some clusters. Since not all the off-diagonal sub-matrices can be well-approximated by the ACA directly, we calculate η for every two clusters in the same level as following:

$$\eta=\frac{\min\{\text{diam t, diam } s\}}{\text{dist}\,(t,s)},$$

where t and s are two clusters; diam t is the diameter of cluster t, so as to cluster s; dist (t,s) is the distance of the centers of the two clusters. When η is smaller than a given value, the corresponding two clusters are admissible. Otherwise, the corresponding two clusters are inadmissible and we will continue to calculate η for the children of the two clusters in the next level.

Assume that the binary tree has two levels. We also assume that every two different clusters in the first level of the tree structure do not satisfy the admissible condition and every

two different clusters in the second level satisfy the admissible condition. Therefore, the coefficient matrix of the BEM can be written as:

$$\mathbf{K} \approx \mathbf{K}^2 = \begin{bmatrix} \begin{bmatrix} \mathbf{K}_1^2 & \mathbf{U}_1^2 \left(\mathbf{V}_1^2\right)^T \\ \mathbf{U}_2^2 \left(\mathbf{V}_2^2\right)^T & \mathbf{K}_2^2 \end{bmatrix} & \begin{bmatrix} \mathbf{U}_{1,1}^2 \left(\mathbf{V}_{1,1}^2\right)^T & \mathbf{U}_{1,2}^2 \left(\mathbf{V}_{1,2}^2\right)^T \\ \mathbf{U}_{2,1}^2 \left(\mathbf{V}_{2,1}^2\right)^T & \mathbf{U}_{2,2}^2 \left(\mathbf{V}_{2,2}^2\right)^T \end{bmatrix} \\ \begin{bmatrix} \mathbf{U}_{3,1}^2 \left(\mathbf{V}_{3,1}^2\right)^T & \mathbf{U}_{3,2}^2 \left(\mathbf{V}_{3,2}^2\right)^T \\ \mathbf{U}_{4,1}^2 \left(\mathbf{V}_{4,1}^2\right)^T & \mathbf{U}_{4,2}^2 \left(\mathbf{V}_{4,2}^2\right)^T \end{bmatrix} & \begin{bmatrix} \mathbf{K}_3^2 & \mathbf{U}_3^2 \left(\mathbf{V}_3^2\right)^T \\ \mathbf{U}_4^2 \left(\mathbf{V}_4^2\right)^T & \mathbf{K}_4^2 \end{bmatrix} \end{bmatrix}. \tag{18}$$

Note that the partition shown in Eq. (18) is different from Eq. (6). The off-diagonal submatrices $\mathbf{U}_1^1 \left(\mathbf{V}_1^1\right)^T$ and $\mathbf{U}_2^1 \left(\mathbf{V}_2^1\right)^T$ of Eq. (6) are also divided in Eq. (18) because the corresponding cluster pairs do not satisfy the admissible condition. For Eq. (18), it is not as easy as Eq. (6) to calculate the inverse of the matrix. Therefore, we present an approach to convert Eqs. (18) to (6).

Assume two clusters t and s in level l are not admissible. t_1 and t_2 are the children of cluster t; s_1 and s_2 are the children of cluster s. Assume the numbers of elements in t_1, t_2, s_1 and s_2 are n_{t_1}, n_{t_2}, n_{s_1} and n_{s_2}, respectively. $\mathbf{K}_{t,s}^l$ is the coefficient matrix capturing the interaction of clusters t and s. $\mathbf{K}_{t,s}^l$ is divided into four submatrices: $\mathbf{K}_{t_1,s_1}^{l+1}$, $\mathbf{K}_{t_1,s_2}^{l+1}$, $\mathbf{K}_{t_2,s_1}^{l+1}$ and $\mathbf{K}_{t_2,s_2}^{l+1}$ based on the division of cluster t and s. Assume t_1 and s_1, t_1 and s_2, t_2 and s_1, and t_2 and s_2 are all admissible. The low-rank approximation of $\mathbf{K}_{t_i,s_j}^{l+1}$ can be written as: $\mathbf{K}_{t_i,s_j}^{l+1} = \mathbf{U}_{t_i,s_j}^{l+1} \left(\mathbf{V}_{t_i,s_j}^{l+1}\right)^{\mathbf{T}}$ $(i, j = 1, 2)$. The dimensions of $\mathbf{U}_{t_i,s_j}^{l+1}$ and $\mathbf{V}_{t_i,s_j}^{l+1}$ are assumed as $n_{t_i} \times k_{t_i,s_j}$ and $n_{s_j} \times k_{t_i,s_j}$, respectively. Therefore, we have the approximation of $\mathbf{K}_{t,s}^l$ as:

$$\begin{aligned} \mathbf{K}_{t,s}^l &= \begin{bmatrix} \mathbf{K}_{t_1,s_1}^{l+1} & \mathbf{K}_{t_1,s_2}^{l+1} \\ \mathbf{K}_{t_2,s_1}^{l+1} & \mathbf{K}_{t_2,s_2}^{l+1} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{t_1,s_1}^{l+1} \left(\mathbf{V}_{t_1,s_1}^{l+1}\right)^{\mathbf{T}} & \mathbf{U}_{t_1,s_2}^{l+1} \left(\mathbf{V}_{t_1,s_2}^{l+1}\right)^{\mathbf{T}} \\ \mathbf{U}_{t_2,s_1}^{l+1} \left(\mathbf{V}_{t_2,s_1}^{l+1}\right)^{\mathbf{T}} & \mathbf{U}_{t_2,s_2}^{l+1} \left(\mathbf{V}_{t_2,s_2}^{l+1}\right)^{\mathbf{T}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{U}_{t_1,s_1}^{l+1} & & \mathbf{U}_{t_1,s_2}^{l+1} & \\ & \mathbf{U}_{t_2,s_1}^{l+1} & & \mathbf{U}_{t_2,s_2}^{l+1} \end{bmatrix} \begin{bmatrix} \left(\mathbf{V}_{t_1,s_1}^{l+1}\right)^{\mathbf{T}} & & & \\ & \left(\mathbf{V}_{t_2,s_1}^{l+1}\right)^{\mathbf{T}} & & \\ & & \left(\mathbf{V}_{t_1,s_2}^{l+1}\right)^{\mathbf{T}} & \\ & & & \left(\mathbf{V}_{t_2,s_2}^{l+1}\right)^{\mathbf{T}} \end{bmatrix} \\ &= \mathbf{U}_{t,s}^l \left(\mathbf{V}_{t,s}^l\right)^{\mathbf{T}} \end{aligned} \tag{19}$$

With Eq. (19), if two clusters t and s are not admissible, the approximation of $\mathbf{K}_{t,s}^l$ is expressed with the low-rank approximation matrices in the lower level. However, directly using Eq. (19) as the approximation of an off-diagonal submatrices $\mathbf{K}_{t,s}^l$ may not be efficient. As we mentioned before, the Sherman–Morrison–Woodbury formula is very efficient when the second dimensions of $\mathbf{U}$ and $\mathbf{V}$ are much smaller than their first dimensions. However, the second dimensions of $\mathbf{U}_{t,s}^l$ and $\mathbf{V}_{t,s}^l$ are $k_{t_1,s_1} + k_{t_1,s_2} + k_{t_2,s_1} + k_{t_2,s_2}$, which might be large. Therefore, in order to improve the efficiency, the randomized interpolative decomposition [47–50] is applied to $\left(\mathbf{V}_{t,s}^l\right)^T$ to find a new approximation of $\mathbf{K}_{t,s}^l$. For convenience, the randomized interpolative decomposition is applied to $\left[\mathbf{V}_{t_1,s_1}^{l+1} \ \mathbf{V}_{t_2,s_1}^{l+1}\right]^{\mathbf{T}}$ and $\left[\mathbf{V}_{t_1,s_2}^{l+1} \ \mathbf{V}_{t_2,s_2}^{l+1}\right]^{\mathbf{T}}$ separately:

$$\begin{aligned} \left[\mathbf{V}_{t_1,s_1}^{l+1} \ \mathbf{V}_{t_2,s_1}^{l+1}\right]^{\mathbf{T}} &= \left[\mathbf{P}_{t_1,s_1}^{l+1} \ \mathbf{P}_{t_2,s_1}^{l+1}\right]^{\mathbf{T}} \left(\mathbf{V}_{s_1}^{l+1}\right)^{\mathbf{T}} \\ \left[\mathbf{V}_{t_1,s_2}^{l+1} \ \mathbf{V}_{t_2,s_2}^{l+1}\right]^{\mathbf{T}} &= \left[\mathbf{P}_{t_1,s_2}^{l+1} \ \mathbf{P}_{t_2,s_2}^{l+1}\right]^{\mathbf{T}} \left(\mathbf{V}_{s_2}^{l+1}\right)^{\mathbf{T}} \end{aligned} \tag{20}$$

where $\mathbf{V}_{s_1}^{l+1}$ and $\mathbf{V}_{s_2}^{l+1}$ are $n_{s_1} \times k_{s_1}$ and $n_{s_2} \times k_{s_2}$ matrices, respectively. Usually, $k_{s_1} < k_{t_1,s_1} + k_{t_2,s_1}$ and $k_{s_2} < k_{t_1,s_2} + k_{t_2,s_2}$. Substituting Eq. (20) into Eq. (19) gives:

$$\begin{aligned} \mathbf{K}_{t,s}^l &= \begin{bmatrix} \mathbf{U}_{t_1,s_1}^{l+1} & & \mathbf{U}_{t_1,s_2}^{l+1} & \\ & \mathbf{U}_{t_2,s_1}^{l+1} & & \mathbf{U}_{t_2,s_2}^{l+1} \end{bmatrix} \begin{bmatrix} \left(\mathbf{V}_{t_1,s_1}^{l+1}\right)^{\mathbf{T}} & & & \\ \left(\mathbf{V}_{t_2,s_1}^{l+1}\right)^{\mathbf{T}} & & & \\ & & \left(\mathbf{V}_{t_1,s_2}^{l+1}\right)^{\mathbf{T}} & \\ & & \left(\mathbf{V}_{t_2,s_2}^{l+1}\right)^{\mathbf{T}} & \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{U}_{t_1,s_1}^{l+1} & & \mathbf{U}_{t_1,s_2}^{l+1} & \\ & \mathbf{U}_{t_2,s_1}^{l+1} & & \mathbf{U}_{t_2,s_2}^{l+1} \end{bmatrix} \begin{bmatrix} \left(\mathbf{P}_{t_1,s_1}^{l+1}\right)^{\mathbf{T}} & & & \\ \left(\mathbf{P}_{t_2,s_1}^{l+1}\right)^{\mathbf{T}} & & & \\ & & \left(\mathbf{P}_{t_1,s_2}^{l+1}\right)^{\mathbf{T}} & \\ & & \left(\mathbf{P}_{t_2,s_2}^{l+1}\right)^{\mathbf{T}} & \end{bmatrix} \begin{bmatrix} \left(\mathbf{V}_{s_1}^{l+1}\right)^{\mathbf{T}} & \\ & \left(\mathbf{V}_{s_2}^{l+1}\right)^{\mathbf{T}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{U}_{t_1,s_1}^{l+1} \left(\mathbf{P}_{t_1,s_1}^{l+1}\right)^{\mathbf{T}} & \mathbf{U}_{t_1,s_2}^{l+1} \left(\mathbf{P}_{t_1,s_2}^{l+1}\right)^{\mathbf{T}} \\ \mathbf{U}_{t_2,s_1}^{l+1} \left(\mathbf{P}_{t_2,s_1}^{l+1}\right)^{\mathbf{T}} & \mathbf{U}_{t_2,s_2}^{l+1} \left(\mathbf{P}_{t_2,s_2}^{l+1}\right)^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \left(\mathbf{V}_{s_1}^{l+1}\right)^{\mathbf{T}} & \\ & \left(\mathbf{V}_{s_2}^{l+1}\right)^{\mathbf{T}} \end{bmatrix} \\ &= \mathbf{U}_{t,s}^{l'} \left(\mathbf{V}_{t,s}^{l'}\right)^{\mathbf{T}} \end{aligned} \tag{21}$$

where the ranks of $\mathbf{U}_{t,s}^{l'}$ and $\mathbf{V}_{t,s}^{l'}$ are $k_{s_1} + k_{s_2}$, and $k_{s_1} + k_{s_2} < k_{t_1,s_1} + k_{t_1,s_2} + k_{t_2,s_1} + k_{t_2,s_2}$. Equation (21) gives a new approximation of $\mathbf{K}_{t,s}^l$ with lower-rank matrices $\mathbf{U}_{t,s}^{l'}$ and $\mathbf{V}_{t,s}^{l'}$. The value of $k_{s_1} + k_{s_2}$ is controlled by the tolerance of the randomized interpolative decomposition. To achieve higher accuracy, we can use smaller tolerance, which could take longer time to calculate $\mathbf{U}_{t,s}^{l'}$ and $\mathbf{V}_{t,s}^{l'}$ and use more computer memory to store them. However, if a larger tolerance is used to achieve higher efficiency, $\mathbf{K}_{t,s}^l$ might not be approximated accurately by $\mathbf{U}_{t,s}^{l'}$ and $\mathbf{V}_{t,s}^{l'}$. A numerical example will be used to show the appropriate choice of the tolerance later.

With Eq. (21), Eq. (18) can be re-written as a HODLR matrix:

$$\mathbf{K} \approx \mathbf{K}^2 = \begin{bmatrix} \begin{bmatrix} \mathbf{K}_1^2 & \mathbf{U}_1^2 \left(\mathbf{V}_1^2\right)^T \\ \mathbf{U}_2^2 \left(\mathbf{V}_2^2\right)^T & \mathbf{K}_2^2 \end{bmatrix} & \mathbf{U}_1^{1'} \left(\mathbf{V}_1^{1'}\right)^T \\ \mathbf{U}_2^{1'} \left(\mathbf{V}_2^{1'}\right)^T & \begin{bmatrix} \mathbf{K}_3^2 & \mathbf{U}_3^2 \left(\mathbf{V}_3^2\right)^T \\ \mathbf{U}_4^2 \left(\mathbf{V}_4^2\right)^T & \mathbf{K}_4^2 \end{bmatrix} \end{bmatrix}.$$

Now, we can decompose the coefficient matrix in the BEM and calculate its inverse as a HODLR matrix.

4.2 Algorithm

The entire algorithm has two steps: assembling and factorization. The assembling step starts with building a binary tree structure to divide all the elements into some clusters. The next step is to check if two clusters in the same level satisfy the admissible condition from the top of the tree structure. If two clusters satisfy the admissible condition, the ACA is used to approximate the coefficient matrix capturing the interaction of these two clusters. Otherwise, the checking process will continue to the children of these two clusters in the next level of the tree structure. If two clusters in the lowest level of the tree structure still do not satisfy the admissible condition, the corresponding coefficient matrix will be calculated directly. This process is similar to the setup step of the ACA BEM. After the low-rank approximations are calculated for the admissible submatrices, we use Eq. (21) to calculate the approximation for the off-diagonal submatrices capturing the interaction of inadmissible clusters.

The next step is factorization. Assume maximum number of tree level is $l_{\max}$ .We start from level $l_{\max}$:

1. Apply the inverse of $\mathbf{K}_{l_{\max}}$ to the right-hand-side. Because $\mathbf{K}_{l_{\max}}$ is calculated directly, LU decomposition is used to calculate its inverse. Go to the upper level.
2. In the upper level $l = 1, 2, \ldots, l_{\max} - 1$, after the inverse matrices of level $l + 1$ are applied to the right-hand-side, the diagonal block of $\mathbf{K}^l$ has the following structure:

$$\begin{aligned}\mathbf{K}_i^l &= \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{U}}_{2i-1}^{l+1'}\left(\mathbf{V}_{2i-1}^{l+1'}\right)^T \\ \tilde{\mathbf{U}}_{2i}^{l+1'}\left(\mathbf{V}_{2i}^{l+1'}\right)^T & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{U}}_{2i-1}^{l+1'} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{U}}_{2i}^{l+1'} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \left(\mathbf{V}_{2i-1}^{l+1'}\right)^T \\ \left(\mathbf{V}_{2i}^{l+1'}\right)^T & \mathbf{0} \end{bmatrix} \\ &= \mathbf{I} + \mathbf{U}_i^l \left(\mathbf{V}_i^l\right)^T \end{aligned} \tag{22}$$

   In order to apply the inverse of $\mathbf{K}_i^l$, the Sherman–Morrison–Woodbury formula is used:

$$\left(\mathbf{K}_i^l\right)^{-1} = \mathbf{I} - \mathbf{U}_i^l \left(\mathbf{I} + \left(\mathbf{V}_i^l\right)^T \mathbf{U}_i^l\right)^{-1} \left(\mathbf{V}_i^l\right)^T \tag{23}$$

3. Since the diagonal blocks at level l are applied to the right-hand-side, the diagonal block at level $l - 1$ also has the above form. Therefore, Sherman–Morrison–Woodbury formula will be used to obtain the inverse of $\mathbf{K}_i^{l-1}$ again. This upward procedure continues until reaching the top level of the tree structure.

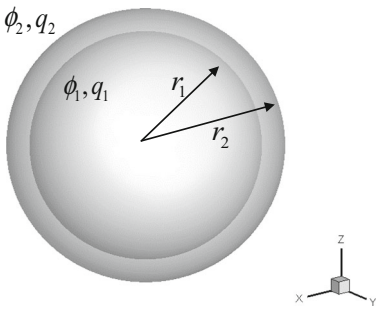

Fig. 2 A spherical shell model

5 Numerical examples

In this section, four different numerical examples are presented to show the accuracy and efficiency of the proposed fast direct solver for the BEM. Solutions of all the example problems are obtained using a single-core computer with Intel Xeon 2.4GHz CPU. No parallel computing is applied for all the examples. Constant triangular elements are used and all integrals are evaluated analytically. The default tolerance of row and column selection in the ACA is set at 10^{-4} . In all the following examples, the L2 error is defined as:

$$\text{L2 error} = \sqrt{\frac{\|\mathbf{x}_{num} - \mathbf{x}_{exact}\|^2}{\|\mathbf{x}_{exact}\|^2}}.$$

5.1 Thin spherical shell model

The first example is a thin spherical shell model (Fig. 2) with the different tolerance ε of the randomized interpolative decomposition. The inner and outer radius of the shell are $r_1 = 0.5$ and $r_2 = 0.6$, respectively. The model is discretized with increasing numbers of elements. Two types of boundary conditions are considered for this model. As the first type of the boundary condition, the potential ϕ is given on the inner surface and the normal derivative of the potential q is given on the outer surface. For the second type of the boundary condition, the potential ϕ are given on both the inner and outer surfaces. Both the conventional BEM solver and the new fast direct solver are used to solve the BEM models.

The L2 errors of these two methods are shown in Table 1. From Table 1, we can see that the tolerance ε of the randomized interpolative decomposition substantially influences the L2 error of the new fast direct BEM solver. When the tolerance is 10^{-5}, the L2 error of the new fast direct solver is very close to that of the conventional BEM. When the tolerance is 10^{-4}, the L2 error of the new fast direct solver is close to that of the conventional BEM only for the model with 2560 elements. When the tolerance is 10^{-3}, the L2 error of the new fast direct solver is much larger than that

Table 1 L2 error of the new fast direct BEM solver for solving the spherical shell model with different tolerances of the interpolative decomposition

N	Tolerance ε for interpolative decomposition	Boundary condition $\phi_1 = 100,\ q_2 = 50$		Boundary condition $\phi_1 = 100,\ \phi_2 = 50$
		L2 error		L2 error
		q (%)	ϕ (%)	q (%)
2560	10^{-3}	2.100	0.107	0.976
	10^{-4}	0.424	0.017	0.462
	10^{-5}	0.413	0.018	0.457
	Conventional BEM	0.412	0.015	0.453
10,240	10^{-3}	3.704	0.207	1.616
	10^{-4}	0.402	0.009	0.205
	10^{-5}	0.156	0.003	0.161
	Conventional BEM	0.153	0.003	0.160
40,960	10^{-3}	7.211	0.250	2.546
	10^{-4}	0.701	0.010	0.209
	10^{-5}	0.091	0.001	0.059
	Conventional BEM	0.056	0.001	0.057

of the conventional BEM. Therefore, the new fast direct solver can obtain accurate results when the tolerance ε of the randomized interpolative decomposition is 10^{-5}. We choose 10^{-5} as the default tolerance of the randomized interpolative decomposition for the following examples if no exception is mentioned.

5.2 Torus model

The second example is a torus model. The distance from the center of the tube to the center of the torus is $R = 5$. The radius of the tube is $r = 2$. The center of the torus is located at the origin of the Cartesian coordinate system. The boundary condition is given as:

$$\begin{aligned}\phi = &\sinh\left(\frac{\sqrt{2}}{4}x\right)\sin\left(\frac{y}{4}\right)\sin\left(\frac{z}{4}\right)\\ &+\sin\left(\frac{x}{4}\right)\sinh\left(\frac{\sqrt{2}}{4}y\right)\sin\left(\frac{z}{4}\right)\\ &+\sin\left(\frac{x}{4}\right)\sin\left(\frac{y}{4}\right)\sinh\left(\frac{\sqrt{2}}{4}z\right),\end{aligned}$$

which is a harmonic function. In this case, the tolerance of the randomized interpolative decomposition is set at 10^{-4} for the models with more than 102,400 elements.

Figure 3 compares the L2 errors of the conventional BEM and the new fast direct BEM solver. From Fig. 3, we can see that the L2 error of the new fast direct solver is very close to that of the conventional BEM, which indicates that the new fast direct solver is almost equally accurate as the conventional BEM. Figure 4 shows the contour plot of the normal derivative of the potential on the surface of the torus with 230,400 elements solved by the new fast direct solver. From Fig. 4, the contour plot of the normal derivative of the potential has symmetric pattern, as expected. Therefore, it is demonstrated that the new fast direct solver can solve the problem as accurately as the conventional BEM (with the traditional direct solver).

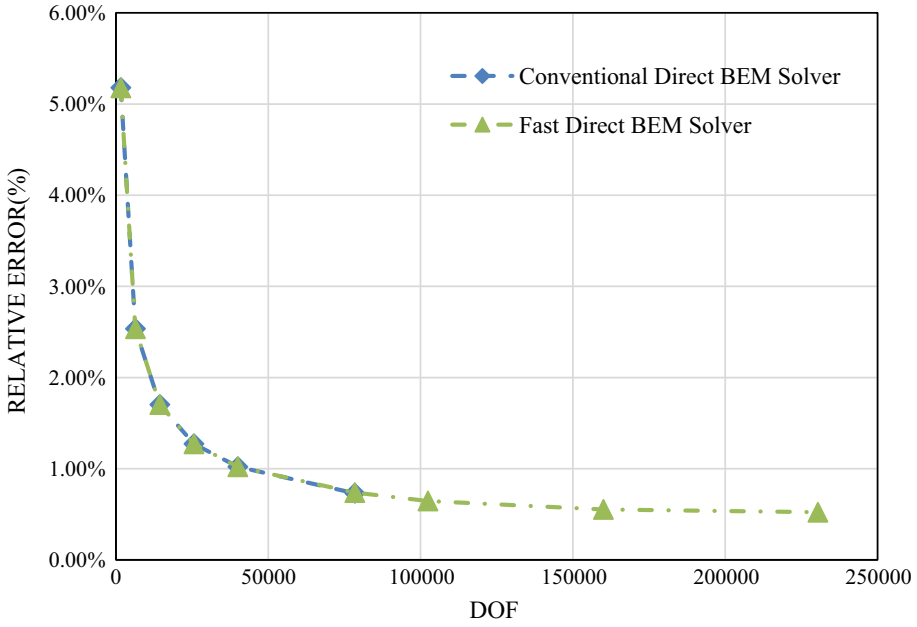

Fig. 3 Relative error of the new fast direct BEM solver and conventional direct BEM solver for solving the torus model

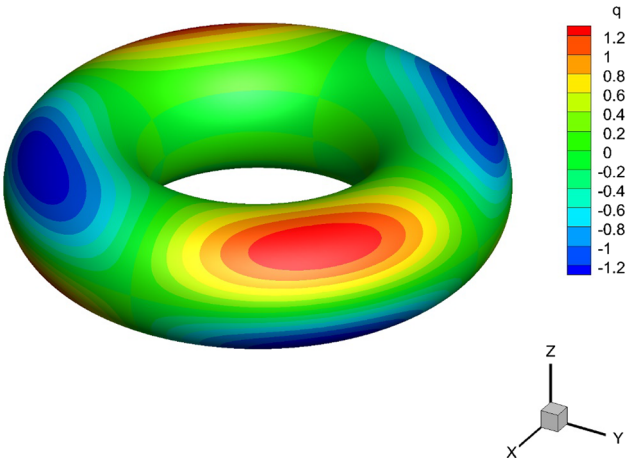

Fig. 4 Contour plots of the normal derivative of potential on the surface of the torus model

Figures 5 and 6 show the CPU time of the new fast direct solver and the conventional BEM for setup and solving the equations, respectively. For the new fast direct solver, the CPU time of setup refers to the CPU time of calculating the right-hand side vector and approximating the coefficient matrix. For the conventional BEM, the CPU time of setup includes the CPU of calculating right-hand side vector and calculating the coefficient matrix directly. The trend lines of the CPU time are also shown. From Fig. 6, we can see that the new fast direct BEM solver uses less CPU time to solve equations for BEM models with more than 10,000 DOFs. The trend line of the new fast direct solver indicates that the complexity of the new fast direct solver for solving equations is nearly $O(N \log N)$, which is much lower than that of the conventional BEM. Therefore, the new fast direct solver will be more advantageous for solving equations as the model size becomes larger. However, the CPU time of these two methods for setup (Fig. 5) are close for this model. We will discuss this phenomenon later. Regarding the total CPU time for solving the problem, the new fast direct BEM solver will be faster than the conventional BEM because of its advantage in solving the system of equations.

Figure 7 shows the used computer memory by using the two methods. The trend lines of used computer memory are also shown. From Fig. 7, the new fast direct solver uses fewer computer memory to obtain the solution for BEM models with more than 10,000 DOFs. The trend line of the fast direct solve indicates that the used computer memory of the new fast direct solver increases almost linearly, which is much slower than the conventional BEM. Therefore, as the model becomes larger, the new fast direct BEM solver should be more efficient than the conventional BEM.

5.3 Long-bar model

A simple long-bar model is considered in this section (Fig. 8). Unlike the torus model, the boundary of the long-bar model is not smooth. The length of the model is $L = 9$. The height and width of the bar are 1. The model is centered at the origin of the Cartesian coordinate system. The length axis of the long-bar model is parallel to the surface $x + y = 0$. In this setup, the boundary condition is given using the following function:

$$\phi = 2x^3 + 2y^3 + 2z^2 - 3\left(x^2y + xy^2\right) - 3\left(x^2z + xz^2\right) - 3\left(y^2z + yz^2\right),$$

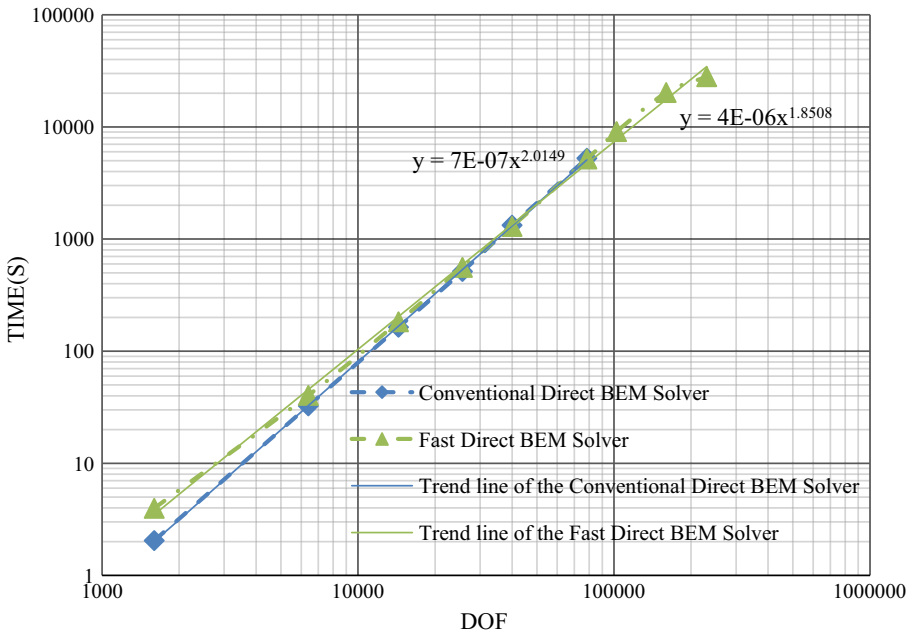

Fig. 5 CPU time of the new fast direct BEM solver and conventional direct BEM solver for setup in the torus model

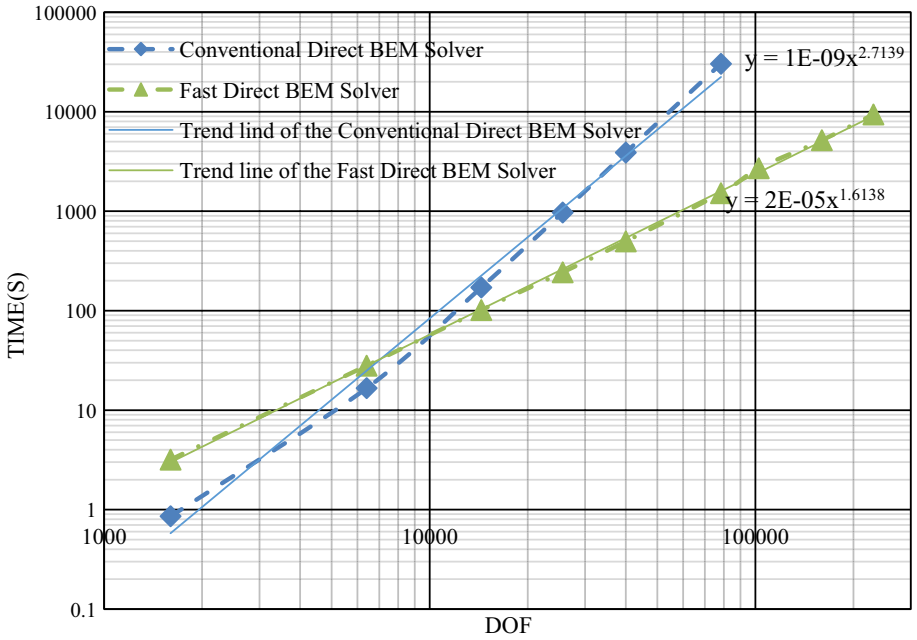

Fig. 6 CPU time of the new fast direct BEM solver and conventional direct BEM solver for solving equations in the torus model

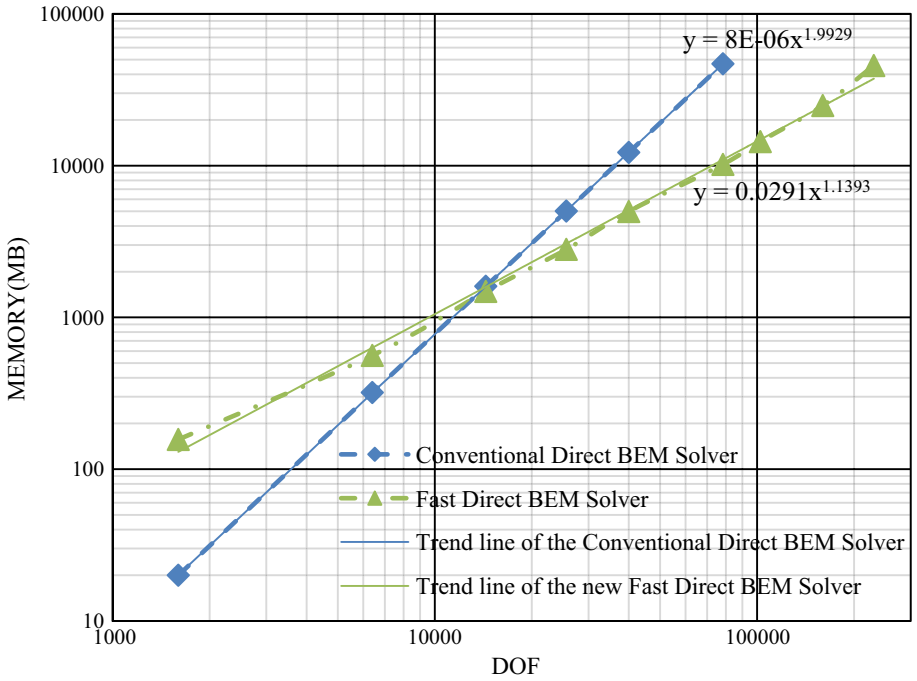

Fig. 7 Computer memory used by the new fast direct BEM solver and conventional direct BEM solver for solving the torus model

which is also a harmonic function.

Table 2 shows the L2 errors of the conventional BEM and the new fast direct solver with increasing numbers of elements. From Table 2, we observe that the L2 errors of the two methods are also close to each other. Figure 9 shows the contour plot of the normal derivative of the potential on the surface of the long-bar model with 68,400 elements. The plots are almost identical and show the same pattern as the number of the elements increases. Therefore, the new fast direct BEM solver also can solve problems with non-smooth boundaries accurately.

5.4 Eleven sphere model

In this example, we use the new fast direct solver to analyze the 11 perfectly conducting sphere model used in Ref. [10] (Fig. 10). The radii of the center large sphere and ten small spheres are 3 and 1, respectively. The ten small spheres are

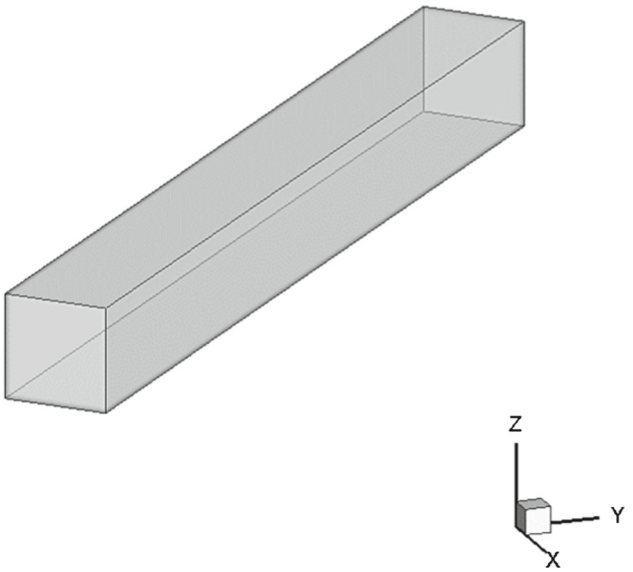

Fig. 8 A long-bar model

Table 2 L2 error of the conventional BEM and the new fast direct BEM solver for solving the long-bar model

Number of elements	L2 error	
	The conventional BEM (%)	The new fast direct solver (%)
1900	9.310	9.308
7600	6.512	6.513
17,100	5.300	5.297
30,400	4.583	4.587
68,400	3.737	3.740

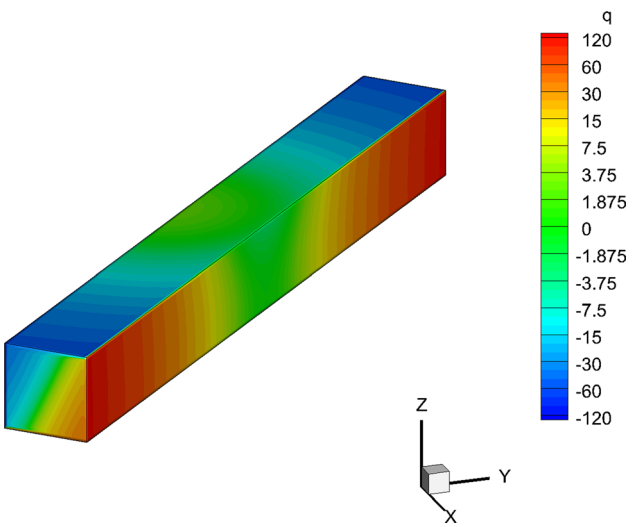

Fig. 9 Contour plot of the normal derivative of the potential on the surface of the long-bar model

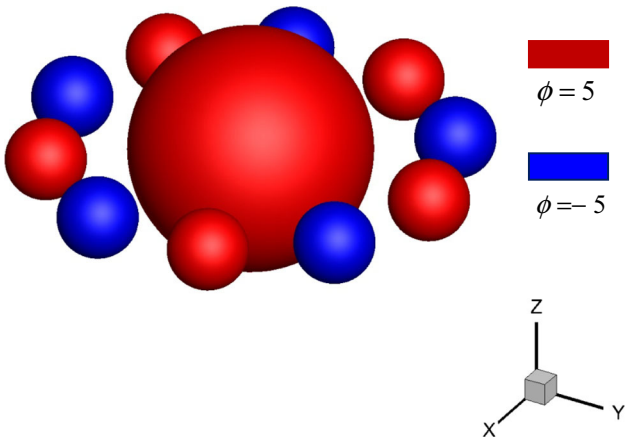

Fig. 10 11 sphere electric conductor model

located uniformly on a circle with radius being 5 and co-centered with the large sphere. A constant electric potential $\phi = 5$ is applied to the center large sphere and five small spheres. The other five small spheres are given a constant electric potential $\phi = -5$. The dielectric constant is set to be 1.

Both the conventional BEM and the new fast direct solver are used to solve the problem. Table 3 shows the maximum and minimum values of the charge density (q) on the surfaces of the spheres. From Table 3, we can see that the maximum and minimum values of the charge density solved by the new fast direct solver gradually converge and have the same first three significant digits as the values solved by using the conventional BEM. The contour plot of the charge density on the surface are shown in Fig. 11 with 4800 elements on each sphere. As the number of elements increases, the plot exhibits almost the same pattern as it should be. Therefore, it is verified that the new fast direct BEM solver can solve larger scale problems accurately.

Regarding the efficiency, the new fast direct solver requires about 1 h 37 min and 22 GB computer memory to solve the problem with 10,800 elements per sphere (Total DOFs = 118,800). However, the conventional BEM needs more than 48 GB computer memory and 4 h CPU time to solve it. Due to limitation of requesting larger computer memory on the cluster, no further run was attempted. This confirms that the new fast direct solver is more efficient than the conventional BEM.

Table 4 shows the maximum and minimum values of the charge density calculated by using the new fast direct solver with different tolerances of selecting row and column in the ACA that is used to compute the submatrices. The tolerance for randomized interpolation decomposition is still 10^{-5}. The model with 4800 elements per sphere is used. The CPU time of the new fast direct solver and the conventional BEM are also shown. As the tolerance of selecting rows and columns

Table 3 Results for the 11-sphere model using the conventional BEM and new fast direct BEM solver

Model		Charge density on the surface of sphere			
		Maximum		Minimum	
Element/sphere	Total DOFs	Conventional BEM	New fast direct solver	Conventional BEM	New fast direct solver
768	8448	10.9927	10.9938	−16.3394	−16.3395
1200	13,200	11.1176	11.1197	−16.4825	−16.4810
1728	19,008	11.1596	11.1646	−16.5618	−16.5621
3072	33,792	11.2314	11.2393	−16.6411	−16.6422
4800	52,800	11.2641	11.2731	−16.6744	−16.6791
10,800	118,800	–	11.3095	–	−16.7254

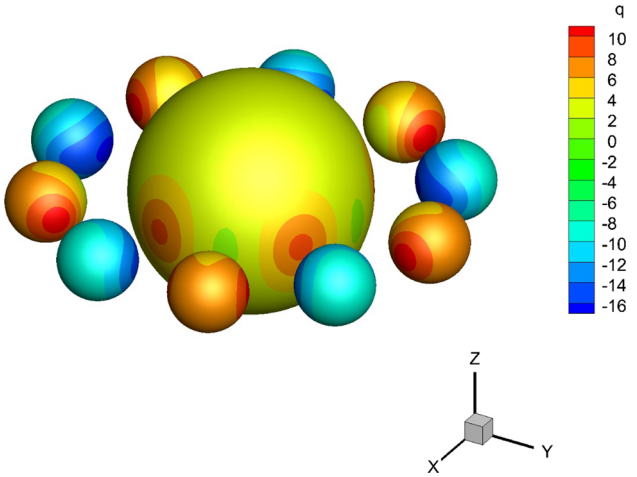

Fig. 11 Contour plot of the charge density on the surface of the 11-sphere model

in the ACA becomes smaller, the maximum and minimum values of the charge density converge and have more identical significant digits compared to the results of the conventional BEM. When the tolerance of row and column selection in ACA is 10^{-8}, the first five significant digits of the maximum and minimum values are identical, and the total CPU time of the new fast direct solver is only about one half of the CPU time of the conventional BEM. It is demonstrated again that the new fast direct solver can obtain accurate results and is more efficient than the conventional BEM for larger scale BEM models.

It is worth mentioning that in Table 4, the new fast direct BEM solver not only saves CPU time in solving the equations, but also saves about 30% CPU time in setup of the equations. However, for the torus model (Sect. 5.2), the CPU time of the new fast direct solver and the conventional BEM for setup are close. This phenomenon indicates that the CPU time of the new fast direct solver for setup depends on the model and boundary conditions. Generally, the more frequently and efficiently the ACA is used in the setup, the more CPU time one can save. For the models requiring larger CPU time for setup, we can use parallel computing to accelerate the setup process. Therefore, the efficiency of the new fast direct solver will not be constrained by the setup.

6 Discussions

In the paper, a new fast direct solver for the BEM is presented. The new fast direct solver is based on the structure of the HODLR matrix. The most important feature of a HODLR matrix is that all the off-diagonal submatrices can be well-approximated by low-rank matrices. Therefore, a HODLR matrix can be factorized into the multiplication of some diagonal-block matrices. The inverse of a HODLR matrix could be easily obtained by using Sherman–Morrison–Woodbury formula. However, most of the coefficient matrices arising from the BEM are not HODLR matrices. In this paper, a simple approach is used to transfer the BEM coefficient matrix to the HODLR matrix. The numerical results show that the new fast direct solver can deliver accurate results with less CPU time and use smaller computer memory compared to the conventional BEM using traditional direct equation solvers.

The accuracy and efficiency of the new fast direct solver depend on the tolerance of the randomized interpolative decomposition. To improve the efficiency without losing the accuracy too much, the adaptive tolerance strategy could be used. In fact, the tolerance is not necessary to be a constant in Eq. (20). We could use different tolerance values based on the size of the matrices. We could also consider to apply other efficient methods to find $\mathbf{V}_{s_1}^{l+1}$ and $\mathbf{V}_{s_2}^{l+1}$ in Eq. (20). The aim is to find the optimal value of k_{s_1} and k_{s_2} so that Eq. (20) is approximated accurately and the Sherman–Morrison–Woodbury formula can be performed efficiently.

To further improve the efficiency of the new fast direct BEM solver, parallel computing can be applied, which is relatively easy to implement and should speed up the matrix formation calculation and the solution readily. Extension of the new fast direct BEM solver to 3-D elastostatic problems

Table 4 Maximum and minimum values of the charge density on the surface of the spheres with different tolerance for the ACA in computing the low rank submatrices

Elements per sphere/total DOFs	Tolerance for the ACA	Max	Min	CPU time (s)		
				Setup	Solution of equations	Total
4800/52,800	10^{-4}	11.2732	−16.6790	806	376	1182
	10^{-6}	11.2648	−16.6741	846	438	1284
	10^{-8}	11.2644	−16.6742	927	522	1449
Conventional BEM		11.2641	−16.6744	1142	2113	3255

should be straightforward. Another promising application of the fast direct solver is to solve problems with multiple right-hand side vectors, such as the capacity extraction for multiple conductors in electrostatic problems [51,52]. After the coefficient matrix is decomposed and the corresponding inverse matrix is calculated, solving the problem with multiple right-hand side vectors is equivalent to perform the matrix-matrix multiplication once for the fast direct solver. In contrast, the fast BEM based on the FMM or ACA with iterative solvers can only handle one right-hand side vector each time, and the system of equations needs to be solved multiple times for multiple right-hand side vectors. Therefore, the fast direct solver can be more efficient for solving the problems with multiple right-hand side vectors.

References

1. Brebbia CA, Dominguez J (1994) Boundary elements: an introductory course. WIT Press, Ashurst
2. Banerjee P, Butterfield R (1994) The boundary element methods in engineering. McGraw-Hill, New York
3. Bonnet M (1999) Boundary integral equation methods for solids and fluids. Meccanica 34(4):301–302. doi:10.1023/a:1004795120236
4. Aliabadi M (2002) The boundary element method: applications in solids and structures, vol 2. Wiley, Chicester
5. Mukherjee S, Mukherjee YX (2005) Boundary methods: elements, contours, and nodes. CRC Press, Boca Raton
6. Liu YJ (2009) Fast multipole boundary element method—theory and applications in engineering. Cambridge University Press, Cambridge
7. Rokhlin V (1985) Rapid solution of integral-equations of classical potential-theory. J Comput Phys 60(2):187–207
8. Greengard L, Rokhlin V (1987) A fast algorithm for particle simulations. J Comput Phys 73(2):325–348
9. Shen L, Liu YJ (2007) An adaptive fast multipole boundary element method for three-dimensional potential problems. Comput Mech 39(6):681–691
10. Liu YJ, Shen L (2007) A dual BIE approach for large-scale modeling of 3-D electrostatic problems with the fast multipole boundary element method. Int J Numer Meth Eng 71(7):837–855. doi:10.1002/nme.2000
11. Bapat MS, Liu YJ (2010) A new adaptive algorithm for the fast multipole boundary element method. Comput Model Eng Sci 58(2):161–184
12. Liu YJ, Nishimura N, Otani Y, Takahashi T, Chen XL, Munakata H (2005) A fast boundary element method for the analysis of fiber-reinforced composites based on a rigid-inclusion model. J Appl Mech 72(1):115–128. doi:10.1115/1.1825436
13. Liu YJ (2006) A new fast multipole boundary element method for solving large-scale two-dimensional elastostatic problems. Int J Numer Meth Eng 65(6):863–881. doi:10.1002/nme.1474
14. Liu YJ (2008) A fast multipole boundary element method for 2D multi-domain elastostatic problems based on a dual BIE formulation. Comput Mech 42(5):761–773. doi:10.1007/s00466-008-0274-2
15. Shen L, Liu YJ (2006) An adaptive fast multipole boundary element method for three-dimensional acoustic wave problems based on the burton-miller formulation. Comput Mech 40(3):461–472. doi:10.1007/s00466-006-0121-2
16. Bapat MS, Shen L, Liu YJ (2009) Adaptive fast multipole boundary element method for three-dimensional half-space acoustic wave problems. Eng Anal Boundary Elem 33(8–9):1113–1123. doi:10.1016/j.enganabound.2009.04.005
17. Wu HJ, Liu YJ, Jiang WK (2012) Analytical integration of the moments in the diagonal form fast multipole boundary element method for 3D acoustic wave problems. Eng Anal Boundary Elem 36(2):248–254. doi:10.1016/j.enganabound.2011.08.004
18. Song JM, Chew WC (1995) Multilevel fast-multipole algorithm for solving combined field integral-equations of electromagnetic scattering. Microw Opt Technol Lett 10(1):14–19
19. Song JM, Lu CC, Chew WC (1997) Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. IEEE Trans Antennas Propag 45(10):1488–1493
20. Darve E, Have P (2003) Fast multipole method for low-frequency electromagnetic scattering. In: Proceedings of the computational fluid and solid mechanics 2003, vols 1 and 2, pp 1299–1302
21. Tsuji P, Ying L (2011) A fast directional algorithm for high-frequency electromagnetic scattering. J Comput Phys 230(14):5471–5487. doi:10.1016/j.jcp.2011.02.013
22. Nishimura N (2002) Fast multipole accelerated boundary integral equation methods. Appl Mech Rev 55(4):299. doi:10.1115/1.1482087
23. Liu YJ, Nishimura N (2006) The fast multipole boundary element method for potential problems: a tutorial. Eng Anal Boundary Elem 30(5):371–381. doi:10.1016/j.enganabound.2005.11.006
24. Liu YJ, Mukherjee S, Nishimura N, Schanz M, Ye W, Sutradhar A, Pan E, Dumont NA, Frangi A, Saez A (2011) Recent advances and emerging applications of the boundary element method. Appl Mech Rev 64(3):1–38. doi:10.1115/1.4005491
25. Hackbusch W (1999) A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices. Computing 62(2):89–108. doi:10.1007/s006070050015
26. Hackbusch W, Khoromskij BN (2000) A sparse $\mathcal{H}$-matrix arithmetic. Computing 64(1):21–47

27. Hackbusch W, Börm S (2002) Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. Computing 69(1):1–35. doi:10.1007/s00607-002-1450-4
28. Börm S, Grasedyck L, Hackbusch W (2003) Introduction to hierarchical matrices with applications. Eng Anal Boundary Elem 27(5):405–422
29. Ambikasaran S (2013) Fast algorithms for dense numerical linear algebra and applications. Stanford University, Stanford
30. Bebendorf M (2000) Approximation of boundary element matrices. Numer Math 86:565–589
31. Bebendorf M, Rjasanow S (2003) Adaptive low-rank approximation of collocation matrices. Computing 70:1–24
32. Bebendorf M, Grzhibovskis R (2006) Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation. Math Methods Appl Sci 29(14):1721–1747. doi:10.1002/mma.759
33. Smajic J, Andjelic Z, Bebendorf M (2007) Fast BEM for eddy-current problems using h-matrices and adaptive cross approximation. IEEE Trans Magn 43(4):1269–1272. doi:10.1109/tmag.2006.890971
34. Maaskant R, Mittra R, Tijhuis A (2008) Fast analysis of large antenna arrays using the characteristic basis function method and the adaptive cross approximation algorithm. IEEE Trans Antennas Propag 56(11):3440–3451. doi:10.1109/tap.2008.2005471
35. Maerten F (2010) Adaptive cross-approximation applied to the solution of system of equations and post-processing for 3D elastostatic problems using boundary element method. Eng Anal Boundary Elem 34:483–491
36. Saad Y (2003) Iterative methods for sparse linear system, 2nd edn. The Society for Industrial and Applied Mathematics, Philadelphia
37. Chen K (2005) Matrix preconditioning techniques and applications. Cambridge University Press, Cambridge
38. Martinsson PG, Rokhlin V (2005) A fast direct solver for boundary integral equations in two dimensions. J Comput Phys 205(1):1–23. doi:10.1016/j.jcp.2004.10.033
39. Corona E, Martinsson P-G, Zorin D (2015) An o(N) direct solver for integral equations on the plane. Appl Comput Harm Anal 38(2):284–317. doi:10.1016/j.acha.2014.04.002
40. Greengard L, Gueyffier D, Martinsson P-G, Rokhlin V (2009) Fast direct solvers for integral equations in complex three-dimensional domains. Acta Numer 18:243–275
41. Ho KL, Greengard L (2012) A fast direct solver for structured linear systems by recursive skeletonization. SIAM J Sci Comput 34(5):A2507–A2532
42. Lai J, Ambikasaran S, Greengard LF (2014) A fast direct solver for high frequency scattering from a large cavity in two dimensions. SIAM J Sci Comput 36(6):B887–B903
43. Ambikasaran S, Darve E (2013) An O(Nlog$N$) fast direct solver for partial hierarchically semi-separable matrices. J Sci Comput 57(3):477–501. doi:10.1007/s10915-013-9714-z
44. Coulier P, Pouransari H, Darve E (2015) The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems. arXiv preprint arXiv:1508.01835
45. Sherman J, Morrison WJ (1950) Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Ann Math Stat 21(1):124–127
46. Woodbury MA (1950) Inverting modified matrices. https://en.wikipedia.org/wiki/Woodbury_matrix_identity
47. Woolfe F, Liberty E, Rokhlin V, Tygert M (2008) A fast randomized algorithm for the approximation of matrices. Appl Comput Harmon Anal 25(3):335–366. doi:10.1016/j.acha.2007.12.002
48. Liberty E, Woolfe F, Martinsson PG, Rokhlin V, Tygert M (2007) Randomized algorithms for the low-rank approximation of matrices. Proc Nat Acad Sci USA 104(51):20167–20172. doi:10.1073/pnas.0709640104
49. Martinsson P-G, Rokhlin V, Tygert M (2011) A randomized algorithm for the decomposition of matrices. Appl Comput Harmon Anal 30(1):47–68. doi:10.1016/j.acha.2010.02.003
50. Halko N, Martinsson PG, Tropp JA (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 53(2):217–288. doi:10.1137/090771806
51. Nabors K, White J (1991) Fastcap: a multipole accelerated 3-D capacitance extraction program. IEEE Trans Comput Aided Des Integr Circuits Syst 10(11):1447–1459
52. Phillips JR, White JK (1997) A precorrected-FFT method for electrostatic analysis of complicated 3-d structures. IEEE Trans Comput Aided Des Integr Circuits Syst 16(10):1059–1072