Journal of Computational and Applied Mathematics 377 (2020) 112904

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A novel fast direct solver for 3D elastic inclusion problems with the isogeometric boundary element method

F.L. Sun, Y.P. Gong, C.Y. Dong *

Department of Mechanics, School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China

ARTICLE INFO

Article history:
Received 17 June 2019
Received in revised form 24 March 2020

Keywords:
Fast direct solver
Isogeometric boundary element method
Accelerated hybrid algorithm
3D elastic inclusion problems

ABSTRACT

We present a novel fast direct solver to simulate 3D large scale elastic inclusion problems. The method combines the isogeometric analysis boundary element method (IGABEM) and the hierarchical off-diagonal low-rank (HODLR) matrix based on non-uniform rational B-splines (NURBS). Hence the 3D geometric surface can be accurately described by the bivariate NURBS basis functions. In order to solve the large scale problems, a stable accelerated algorithm is used to approximate the off-diagonal submatrices by low-rank matrices. Based on the accelerated algorithm, a hybrid approximation algorithm consisting of singular value decomposition (SVD) and adaptive cross approximation (ACA) is proposed to solve the 3D elastic inclusion problems. The validity and accuracy of the method are verified by testing the four methods. Among the numerical results obtained from the four methods, the method proposed in this paper uses less CPU time and storage space to obtain accurate results.

1. Introduction

To understand the mechanical behavior of heterogeneous materials, we need to investigate the interaction between different components such as inclusions and matrix. There are various methods to solve the inclusion problems, including analytical methods [1,2] and numerical methods. Due to the limitation of analytical methods, numerical methods have attracted extensive attention, such as finite element method (FEM) [3,4], volume integral equation method [5], boundary element method (BEM) [6], etc. The FEM has been applied to analyze various inclusion problems, but its main disadvantage is that when the geometric structure changes, the elements need to be reconstructed in matrix and inclusion. The BEM is an alternative numerical method to deal with the inclusion problems, which has many advantages, such as (1) only discretizes the boundary of the studied problems; (2) easy to solve the problems of infinite domain; (3) applicable to solving crack problems, etc.

However, as a numerical method, the BEM, like the FEM, also has discretization error caused by geometric approximation. Therefore, the precision of numerical solution in numerical calculation is affected by geometric discrete error. The isogeometric analysis (IGA) [7] is an effective method to solve the above problem, in which the geometric shapes are accurately described by NURBS basis functions. The IGABEM was developed by Simpson et al. [8], which combines the advantages of IGA and BEM. Thus, the IGABEM is especially suitable for the numerical simulation of heterogeneous materials containing inclusions of complex shapes. But the IGABEM also inherits the weaknesses of the conventional BEM, such as dense and asymmetric coefficient matrix, which makes the IGABEM unsuitable for large scale problems. The reason mainly lies in the storage of dense matrix and the solution of the system of linear equations. If the number of equations

* Corresponding author.
E-mail address: cydong@bit.edu.cn (C.Y. Dong).

https://doi.org/10.1016/j.cam.2020.112904
0377-0427/

of the system of linear equations is N, N^2 units of memory and $O(N^3)$ arithmetic operations are required to directly solve the system of linear equations. Therefore, a very large number N will result in a dramatic decrease in the efficiency of the solution.

Many methods have been developed to overcome the storage and CPU time problems caused by large dense matrices. For example, in order to make full use of the tensor structure of the B-spline, a fast algorithm of 3D Galerkin matrix based on the low-rank separable tensor decomposition of integral kernels was proposed [9]. There are too many fast algorithms to list them all. If we consider the solution of the system of linear equations, these methods can be summarized into two categories: fast algorithms of iterative methods and fast algorithms of direct solver methods. The fast algorithms of iterative methods usually include fast multipole method (FMM) [10,11], panel clustering methods [12], wavelet based methods [13] and hierarchical matrix ($\mathcal{H}$-matrices) [14–16], etc. In these methods, the FMM significantly accelerates the solution of the integral equations and makes the computational complexity of BEM close to $O(N)$. However, in the implementation of FMM, the difficulty is that the kernel functions of the boundary integral equations need to be approximated by explicitly multipole expansion form. It is well known that the kernel functions of boundary integral equations are different for different problems, and even for the same problem, the fundamental solutions of two and three dimensional problems, such as Laplace's equation and Helmholtz equation, are different. Hence, the multipole expansions of the kernel functions are cumbersome. Another option is the adaptive cross approximation (ACA) algorithm proposed in [17,18], which is a pure algebraic matrix compression method independent of the physical background. Therefore, the $\mathcal{H}$-matrix with ACA becomes popular, especially in the application of BEM. When the coefficient matrix of the problems is ill-posed, the fast algorithm based on iteration methods has a poor convergence rate. In order to avoid dealing with the iterative problems, some research groups [19–23] focus on the study of the fast direct solver. For example, some studies [19–21,24] used the concept of hierarchical off-diagonal low-rank (HODLR) matrix recursively update the inverse of the coefficient matrix using the Sherman–Morrison–Woodbury formula [25].

In this paper, based on the HODLR matrix, a novel fast direct solver for the 3D elastic inclusion problems with IGABEM is proposed. As is well known, all the off-diagonal submatrices of the HODLR matrix are low-rank matrices, which can be obtained by the ACA algorithm. However, when the ACA is directly applied to the decomposition of the large scale off-diagonal submatrices, the efficiency is low due to the long CPU time and large rank. In order to solve this problem, we use the accelerated algorithm of the low-rank approximation of large scale matrix to improve the compression efficiency. Based on the accelerated algorithm, the accelerated hybrid algorithm is introduced to deal with the elastic problems. Since the fundamental solution of the elastic problems is a tensor, we use singular value decomposition (SVD) to approximate the bottom submatrices generated by the accelerated algorithm and ACA to recompress the obtained low-rank matrices. When the IGABEM is used, there are also various singular integrals, such as weak singular integrals, strong singular integrals and hyper-singular integrals. Many methods have been developed to deal with the above problem, such as power series expansion method [26] and weighted quadrature method [27–29]. The power series expansion method, which can be used to evaluate singular integrals of different orders, has been extended to the IGABEM in [30]. In [27,28], the weighted quadrature method was proposed to deal with the singular integrals on the boundary of symmetric Galerkin boundary element method and IGABEM, respectively.

The structure of the paper is organized as follows. Section 2 presents the boundary integral equation for elastic inclusion problems. Section 3 introduces the matrix decomposition methods and the HODLR matrix. Section 4 outlines the application of the fast direct solver based on the IGABEM and proposes the hybrid algorithm to raise the computational efficiency. In Section 5, we use the adaptive integral method to deal with the nearly singular integrals of stress integral equations. In Section 6, four examples are given to illustrate the distribution of local stress, and the accuracy and effectiveness of the proposed method are verified. Finally, we conclude our work in Section 7.

2. Isogeometric boundary integral equation

2.1. The boundary integral equation for inclusion model

To illustrate the proposed numerical method, a simple infinite domain model with inclusions will be considered in this work. As shown in Fig. 1, the matrix ($\overline{\Omega}$) containing some inclusions (Ω^I, $I = 1, 2,\ldots, n$) is subjected to a remote stress field. The interface of the matrix and the Ith inclusion is denoted by Γ_I. The boundary Γ is taken as the sum of the interface boundaries (Γ_I), namely $\Gamma = \sum_{I=1}^{n} \Gamma_I$. Here, a perfectly bonded interface between the matrix and each inclusion is assumed.

For any point $\boldsymbol{y} \in \Gamma$, the interface displacement boundary integral equation can be expressed as [6]

$$c_{ij}(\boldsymbol{y})u_j(\boldsymbol{y}) = u_i^0(\boldsymbol{y}) + \int_\Gamma t_j(\boldsymbol{x})U_{ij}(\boldsymbol{x},\boldsymbol{y})d\Gamma(\boldsymbol{x}) - \int_\Gamma u_j(\boldsymbol{x})T_{ij}(\boldsymbol{x},\boldsymbol{y})d\Gamma(\boldsymbol{x}) \tag{1}$$

where $i, j = 1, 2$ and 3. $\boldsymbol{x}$ is the field point on the inclusion–matrix interface Γ. $u_i^0(\boldsymbol{y})$ is the ith displacement component caused by remote stresses in an infinite homogeneous isotropic elastic matrix. u_j and t_j represent the displacement and traction components, respectively. $c_{ij}(\boldsymbol{y}) = 1/2$ on smooth surface. $U_{ij}(\boldsymbol{x},\boldsymbol{y})$ and $T_{ij}(\boldsymbol{x},\boldsymbol{y})$ are displacement and traction fundamental solutions, respectively, which are given as

$$U_{ij}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{16\pi G(1-\upsilon)r}\left[(3-4\upsilon)\delta_{ij} + r_{,i}r_{,j}\right] \tag{2a}$$

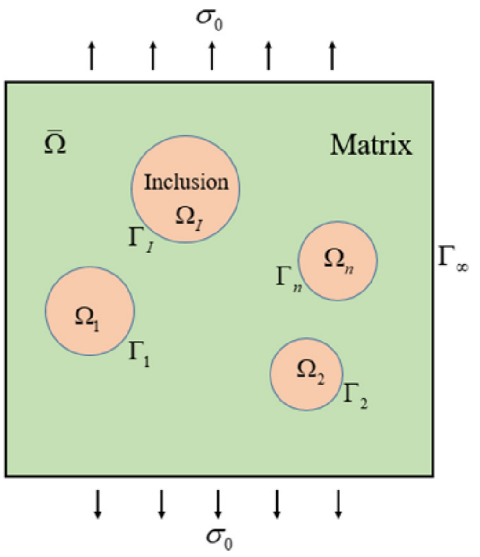


Fig. 1. Inclusion model.

$$T_{ij}(\boldsymbol{x},\boldsymbol{y}) = -\frac{1}{8\pi(1-v)r^2}\left\{\frac{\partial r}{\partial \boldsymbol{n}}\left[(1-2v)\delta_{ij}+3r_{,i}r_{,j}\right]-(1-2v)(r_{,i}n_j-r_{,j}n_i)\right\} \tag{2b}$$

where G and v are shear modulus and Poisson's ratio, respectively. $r_{,i}=\frac{\partial r(\boldsymbol{x},\boldsymbol{y})}{\partial x_i(\boldsymbol{x})}$ in which $r=\|\boldsymbol{x}-\boldsymbol{y}\|$. $\frac{\partial r}{\partial \boldsymbol{n}}=r_{,i}n_i$ in which n_i is the ith component of the unit outward normal vector $\boldsymbol{n}$ to the boundary surface at point $\boldsymbol{x}$. δ_{ij} is the Kronecker delta.

For the Ith isotropic inclusion, when the source point $\boldsymbol{y}$ is on the Ith inclusion–matrix interface Γ_I, the corresponding displacement boundary integral equation can be expressed as follows [31]

$$c^I_{ij}(\boldsymbol{y})u^I_j(\boldsymbol{y}) = \int_{\Gamma_I} t^I_j(\boldsymbol{x})U^I_{ij}(\boldsymbol{x},\boldsymbol{y})d\Gamma(\boldsymbol{x}) - \int_{\Gamma_I} u^I_j(\boldsymbol{x})T^I_{ij}(\boldsymbol{x},\boldsymbol{y})d\Gamma(\boldsymbol{x}) \tag{3}$$

Assume that the Poisson's ratio of the matrix is the same as the Poisson's ratio of inclusions. From Eqs. (2a) and (2b), the following relationships can be found [6,32]

$$U^I_{ij} = \frac{G}{G^I}U_{ij} \tag{4a}$$

$$T^I_{ij} = T_{ij} \tag{4b}$$

Then, based on the interface conditions and Eqs. (4a) and (4b), we can obtain the following displacement boundary integral equation for one inclusion [6]

$$c_{ij}(1+\frac{G^I}{G})u_j(\boldsymbol{y}) = u^0_i(\boldsymbol{y}) - \int_\Gamma u_j(\boldsymbol{x})(1-\frac{G^I}{G})T_{ij}(\boldsymbol{x},\boldsymbol{y})d\Gamma(\boldsymbol{x}) \tag{5}$$

2.2. NURBS discretization

In this section, we focus on the numerical implementation of IGABEM with NURBS basis functions in the 3D heterogeneous elastic problems. Therefore, an overview of NURBS basis functions is given below. More details about NURBS basis function can be found in [7,8].

Two non-decreasing knot vectors along the directions ξ and η are $\Xi=\{\xi_1,\xi_2,\ldots,\xi_{n+p+1}\}$ and $\mathcal{H}=\{\eta_1,\eta_2,\ldots,\eta_{m+q+1}\}$, respectively, where p and q are the orders of univariate basis functions, and n and m represent the total numbers of control points along the directions ξ and η, respectively. The univariate NURBS basis functions in the ξ direction with the positive weight ω_i can be defined as

$$R_{i,p}(\xi) = N_{i,p}(\xi)\omega_i \Big/ \left(\sum_{j=1}^{n} N_{j,p}(\xi)\omega_j\right) \tag{6}$$

Then the bivariate NURBS basis functions can be constructed by the tensor product of two univariate NURBS basis functions as follows

$$R_{i,j}^{p,q}(\xi,\eta)=\frac{N_{i,p}(\xi)M_{j,q}(\eta)\omega_{i,j}}{\sum_{\hat{i}=1}^{n}\sum_{\hat{j}=1}^{m}N_{i,p}(\xi)M_{\hat{j},q}(\eta)\omega_{\hat{i},\hat{j}}} \tag{7}$$

where $N_{i,p}$ and $M_{j,q}$ are the pth and qth order B-spline basis functions defined in the ξ and η directions, respectively. $\omega_{i,j}$ is the weight value associated with bidirectional net of control point $\boldsymbol{P}_{i,j}$. So the NURBS geometric surface in the parametric domain $[\xi_1, \xi_{n+p+1}]\times[\eta_1, \eta_{m+q+1}]$ can be defined as follows

$$\boldsymbol{x}(\xi,\eta)=\sum_{i=1}^{n}\sum_{j=1}^{m}R_{i,j}^{p,q}(\xi,\eta)\boldsymbol{P}_{i,j} \tag{8}$$

In the implementation of IGABEM, the physical quantities are approximated by the same bivariate basis functions that are used to model the boundary surface. Based on the knot vectors, the integrals over the entire boundary of the computed model can be divided into elemental integrals over isogeometric elements. In fact, due to the adoption of NURBS, the element in conventional BEM has been replaced by knot span. In the parametric space, the isogeometric elements are defined as non-zero knot intervals $[\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$, where $\xi_i, \xi_{i+1} \in \Xi$ and $\eta_j, \eta_{j+1} \in \mathcal{H}$. Using the isogeometric elements, the geometry boundary, displacement and traction components on the Γ_e, which is an element of boundary Γ, are mapped to the corresponding knot intervals $[\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}]$ as follows [30]

$$\boldsymbol{x}(\xi,\eta)=\sum_{b=1}^{(p+1)(q+1)}R_b(\xi,\eta)\boldsymbol{x}_b \tag{9a}$$

$$u_i(\xi,\eta)=\sum_{b=1}^{(p+1)(q+1)}R_b(\xi,\eta)d_i^b \tag{9b}$$

$$t_i(\xi,\eta)=\sum_{b=1}^{(p+1)(q+1)}R_b(\xi,\eta)q_i^b \tag{9c}$$

in which $\boldsymbol{x}_b$ denotes the set of control point coordinates and $\boldsymbol{x} = (x, y, z)$ is the location of the physical surface corresponding to the spatial coordinates ξ, η in parametric space. d_i and q_i are the local displacement and traction component parameters at the corresponding control points, respectively.

Then, the integral over the isogeometric boundary element can be computed by the standard Gauss–Legendre rule. The conversion relationship is as follows

$$\xi=(1-\hat{\xi})\xi_i/2+(1+\hat{\xi})\xi_{i+1}/2,\quad \xi\in[\xi_i,\xi_{i+1}] \tag{10a}$$

$$\eta=(1-\hat{\eta})\eta_j/2+(1+\hat{\eta})\eta_{j+1}/2,\quad \eta\in\left[\eta_j,\eta_{j+1}\right] \tag{10b}$$

where $\hat{\xi} \in [-1, 1]$ and $\hat{\eta} \in [-1, 1]$. If there are NE elements, the boundary integral equation (5) at collocation point $\boldsymbol{s} = (\hat{\xi}_s, \hat{\eta}_s)$ can be discretized as follows

$$c_{ij}(1+\frac{G^I}{G})\sum_{J=1}^{(p+1)(q+1)}R_J(\hat{\xi}_s,\hat{\eta}_s)d_j^J=u_i^0-\sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}\hat{T}_{ij}^J d_j^J \tag{11}$$

and

$$\hat{T}_{ij}^J=\int_{-1}^{1}\int_{-1}^{1}(1-\frac{G^I}{G})T_{ij}(\boldsymbol{s},\boldsymbol{x}(\hat{\xi},\hat{\eta}))R_J(\hat{\xi},\hat{\eta})J(\hat{\xi},\hat{\eta})d\hat{\xi}d\hat{\eta} \tag{12}$$

where $J(\hat{\xi},\hat{\eta}) = J_1(\hat{\xi},\hat{\eta})J_2(\xi,\eta)$ is the Jacobian in which $J_1(\hat{\xi},\hat{\eta}) = (\xi_{i+1}-\xi_i)(\eta_{j+1}-\eta_j)/4$ and $J_2(\xi,\eta) = \sqrt{\left(\frac{\partial y}{\partial\xi}\frac{\partial z}{\partial\eta}-\frac{\partial y}{\partial\eta}\frac{\partial z}{\partial\xi}\right)^2+\left(\frac{\partial z}{\partial\xi}\frac{\partial x}{\partial\eta}-\frac{\partial z}{\partial\eta}\frac{\partial x}{\partial\xi}\right)^2+\left(\frac{\partial x}{\partial\xi}\frac{\partial y}{\partial\eta}-\frac{\partial x}{\partial\eta}\frac{\partial y}{\partial\xi}\right)^2}$.

The matrix form of Eq. (11) with respect to boundary condition can be written as

$$\hat{\boldsymbol{H}}\boldsymbol{d}=\boldsymbol{u}^0 \tag{13}$$

where $\boldsymbol{u}^0$ is the known nodal vector composed of displacement components u_i^0 in Eq. (11). $\hat{\boldsymbol{H}}$ is the related coefficient matrix from Eq. (11). By solving Eq. (13), we can get the displacement $\boldsymbol{d}$ of the control points on the side of the matrix at the inclusion–matrix interface. Once the interface displacements are obtained by Eq. (13), the tractions at the corresponding control points can be computed by the integral equation in Eq. (1). The discretized form of Eq. (1) is expressed as follows

$$c_{ij}\sum_{J=1}^{(p+1)(q+1)}R_J(\hat{\xi}_s,\hat{\eta}_s)d_j^J=u_i^0+\sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}U_{ij}^J q_j^J-\sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}T_{ij}^J d_j^J \tag{14}$$

where

$$U_{ij}^J = \int_{-1}^{1}\int_{-1}^{1} U_{ij}(\boldsymbol{s}, \boldsymbol{x}(\hat{\xi}, \hat{\eta}))R_J(\hat{\xi}, \hat{\eta})J(\hat{\xi}, \hat{\eta})d\hat{\xi}d\hat{\eta} \tag{15a}$$

$$T_{ij}^J = \int_{-1}^{1}\int_{-1}^{1} T_{ij}(\boldsymbol{s}, \boldsymbol{x}(\hat{\xi}, \hat{\eta}))R_J(\hat{\xi}, \hat{\eta})J(\hat{\xi}, \hat{\eta})d\hat{\xi}d\hat{\eta} \tag{15b}$$

Introducing the obtained displacement $\boldsymbol{d}$ into Eq. (14), the following matrix form is obtained

$$\boldsymbol{Uq} = \boldsymbol{u}^0 + \boldsymbol{H}\bar{\boldsymbol{d}} \tag{16}$$

where $\bar{\boldsymbol{d}}$ is the obtained displacement vector. Then, Eq. (16) can be rewritten as

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{17}$$

where $\boldsymbol{A} = \boldsymbol{U}.\boldsymbol{x}$ is the unknown traction $\boldsymbol{q}$ of the control points. $\boldsymbol{b} = \boldsymbol{u}^0 + \boldsymbol{H}\bar{\boldsymbol{d}}$ is the known quantity.

In numerical implementation, the power series expansion method [26] is utilized to calculate both the weakly singular integrals and strongly singular integrals existing in Eqs. (11) and (14). In IGA, if we use the initial control parameters to solve the problem under consideration, the accuracy of the displacements and tractions may not meet our requirement. Therefore, this paper adopts the *h*-refinement scheme proposed in [7] to refine the isogeometric elements. The collocation points are generated by the Greville abscissae definition [33].

3. Matrices decomposition and HODLR matrices

3.1. Low-rank matrices decomposition

Given a matrix $\boldsymbol{O} \in \mathbb{R}^{m\times n}$, a low-rank approximation matrix $\boldsymbol{O}_k$ whose rank is $k < \min(m,n)$ is built by a factorization form, namely $\boldsymbol{O}_k = \boldsymbol{UV}$, in which the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are of dimensions $m \times k$ and $k \times n$, respectively. Then $\boldsymbol{O} = \boldsymbol{O}_k + \boldsymbol{R}_k$ where $\boldsymbol{R}_k$ is the residual matrix. There are many techniques which can decompose the matrix $\boldsymbol{O}$ to compute the factored matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. In the course of this section, SVD and ACA will be introduced. The details are as follows.

3.1.1. The singular value decomposition

Assume matrix $\boldsymbol{O}$ is decomposed by using the SVD [34], the positive and non-increasing singular values of matrix $\boldsymbol{O}$ are located on the diagonal of matrix $\boldsymbol{W}_r \in \mathbb{R}^{r\times r}$, namely, $\boldsymbol{W}_r = \text{diag}\,(w_1, w_2, \ldots, w_r)$. $\boldsymbol{U}_r \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V}_r \in \mathbb{R}^{n\times r}$ are orthonormal matrices. In the current numerical computation, the truncated SVD is used to obtain a low-rank matrix $\boldsymbol{O}_k$. Given a threshold ε_{svd} and traversing the singular values, we truncate it when $w_{k+1} < \varepsilon_{svd}$. Then the low rank matrix $\boldsymbol{O}_k = \boldsymbol{U}_k\boldsymbol{W}_k\boldsymbol{V}_k^*$ is obtained. Finally, the factored matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are given as

$$\boldsymbol{U} = \boldsymbol{U}_k\sqrt{\boldsymbol{W}_k} \text{ and } \boldsymbol{V} = \sqrt{\boldsymbol{W}_k}\boldsymbol{V}_k^* \tag{18}$$

where $\sqrt{\boldsymbol{W}_k} = diag(\sqrt{w_1}, \sqrt{w_2}, \ldots, \sqrt{w_k})$.

It is well known that the truncated SVD is optimal for any given rank, which will result in the minimum residual matrix $\boldsymbol{R}_k$ in the Frobenius norm $\|\cdot\|_F$. In other words, for a given accuracy, the truncated SVD can find the lowest rank. However, the drawbacks of the SVD are the storage requirement and the computational complexity. In the SVD implementation, whether $\boldsymbol{O}$ is dense or not, the orthonormal matrices $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ are usually dense. And the cost of SVD for matrix $\boldsymbol{O} \in \mathbb{R}^{m\times n}$ is $14mn^2 + 8n^3$ complex operations [35]. This means the computational cost is cubic, which prevents the application of large scale problems. But for low-rank matrix, the complex operations are reduced to $k^2(m+n)$.

3.1.2. The adaptive cross approximation algorithm

As an alternative to the SVD, the ACA is an efficient technique to factor the low-rank matrix. Instead of building the whole matrix beforehand, we choose some rows and columns from the original matrix to form the following formula [18]

$$\boldsymbol{O}_k = \sum_{i=1}^{k} \boldsymbol{u}_i\boldsymbol{v}_i^* = \boldsymbol{UV} \tag{19}$$

where $\boldsymbol{u}_i \in \mathbb{R}^m$ and $\boldsymbol{v}_i \in \mathbb{R}^n$. So, for any matrix $\boldsymbol{O} \in \mathbb{R}^{m\times n}$, we only need $k \times (m+n)$ terms to store the size of the matrix. The $\varepsilon_{ACA} > 0$ is a given parameter which controls the number of required rows and columns of the original matrix.

There are two different ACA algorithms: fully pivoted ACA and partially pivoted ACA. The detailed description of these two algorithms is given in [18]. For the fully pivoted ACA algorithm, in order to generate the low-rank matrix, the number of operations is $O(kmn)$ and the storage requirement is $O(nm)$. That is, the fully pivoted ACA algorithm is not a satisfying method for large scale matrices. But we can see that the fully pivoted ACA algorithm is simple and the stop criterion can be given explicitly. Compared with the fully pivoted ACA algorithm, the partially pivoted ACA algorithm is called a true fast algorithm in practice. But the algorithm is complex and only an appropriate stopping criterion is given.

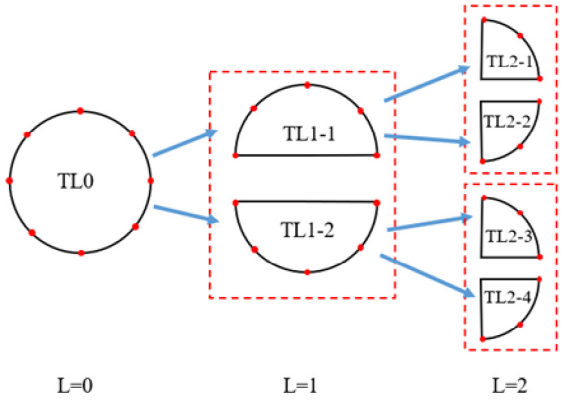


Fig. 2. A binary tree of 2D sketch.

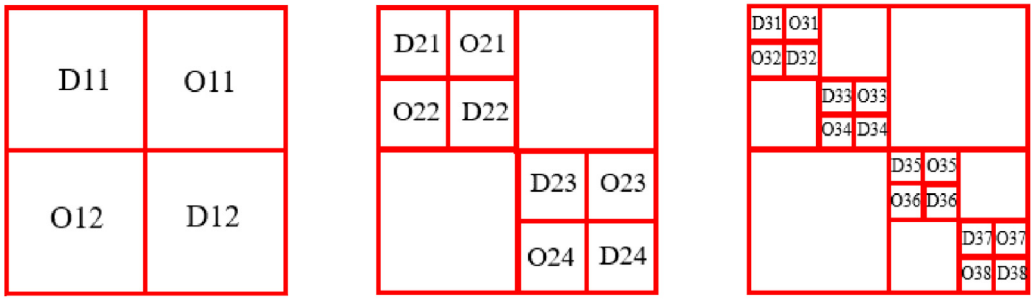


Fig. 3. HODLR matrices for three levels.

3.2. The HODLR matrix

As mentioned in the introduction, the HODLR matrix whose all off-diagonal submatrices can be approximated by low-rank matrices is a kind of H-matrix. Some details about the HODLR matrix are presented in this section.

For a matrix $\boldsymbol{O} \in \mathbb{R}^{m\times n}$, the one-level HODLR matrix is defined as follows [19,20]

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{D}_{11} & \boldsymbol{U}_{11}\boldsymbol{V}_{11} \\ \boldsymbol{U}_{12}\boldsymbol{V}_{12} & \boldsymbol{D}_{12} \end{bmatrix} \tag{20}$$

where $\boldsymbol{D}_{11} \in \mathbb{R}^{m_1\times n_1}$ and $\boldsymbol{D}_{12} \in \mathbb{R}^{m_2\times n_2}$ are the diagonal submatrices. The multiplication form of two matrices $\boldsymbol{U}_{1i}\boldsymbol{V}_{1i}$ $(i = 1, 2)$ is the low-rank approximation of two off-diagonal submatrices, respectively, where $\boldsymbol{U}_{1i} \in \mathbb{R}^{m_i\times k_i}$ and $\boldsymbol{V}_{1i} \in \mathbb{R}^{k_i\times(n-n_i)}$. The first subscript is the number of level. To generate a two-level HODLR matrix, two diagonal submatrices $\boldsymbol{D}_{11}$ and $\boldsymbol{D}_{12}$ in Eq. (20) should be divided into four submatrices, respectively. A two-level HODLR matrix is shown as

$$\boldsymbol{O} = \begin{bmatrix} \begin{bmatrix} \boldsymbol{D}_{21} & \boldsymbol{U}_{21}\boldsymbol{V}_{21} \\ \boldsymbol{U}_{22}\boldsymbol{V}_{22} & \boldsymbol{D}_{22} \end{bmatrix} & \boldsymbol{U}_{11}\boldsymbol{V}_{11} \\ \boldsymbol{U}_{12}\boldsymbol{V}_{12} & \begin{bmatrix} \boldsymbol{D}_{23} & \boldsymbol{U}_{23}\boldsymbol{V}_{23} \\ \boldsymbol{U}_{24}\boldsymbol{V}_{24} & \boldsymbol{D}_{24} \end{bmatrix} \end{bmatrix} \tag{21}$$

Construct recursively until the required l-level is reached. Therefore, the kth diagonal block of l-level HODLR matrix can be written as

$$\begin{bmatrix} \boldsymbol{D}_{l(2k-1)} & \boldsymbol{U}_{l(2k-1)}\boldsymbol{V}_{l(2k-1)} \\ \boldsymbol{U}_{l(2k)}\boldsymbol{V}_{l(2k)} & \boldsymbol{D}_{l(2k)} \end{bmatrix} \tag{22}$$

Assume that the inverse of all diagonal matrices exists and off-diagonal matrices can be well approximated by low-rank matrices. To calculate the inverse of HODLR matrix using the Sherman–Morrison–Woodbury formula [25], the HODLR matrix should be factored into the multiplication of several diagonal block matrices. The readers can refer to the Refs. [21] and [24] for the algorithms of the one-level and multi-level schemes. We focus on the fast algorithm with the IGABEM.

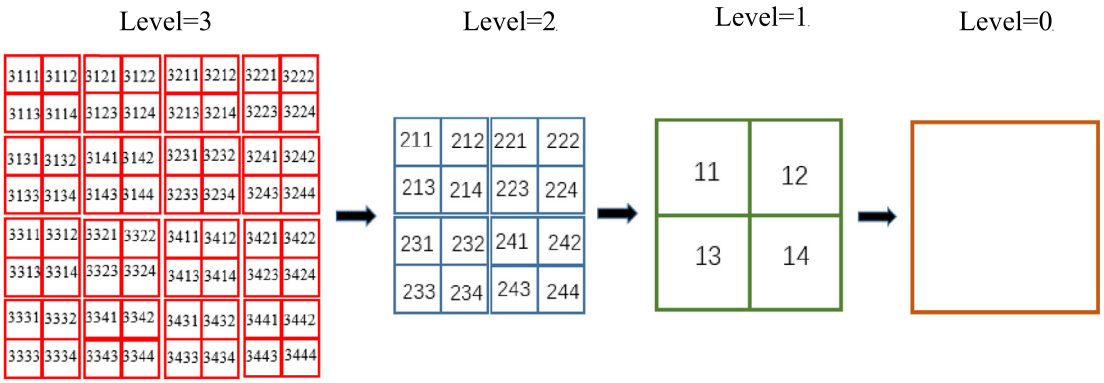


Fig. 4. The accelerated algorithm for three levels.

4. The application of fast algorithm with IGABEM

We can now relate the IGABEM in Section 2 to the HODLR matrix. In order to use the above HODLR matrix to solve the system of linear equations, the hierarchical scheme will be obtained by using the geometric structure. In this work, the HODLR matrix is constructed through a binary tree. In order to better describe the process, a 2D circle model is used and its knot vector is defined as $\Xi = \{0,0,0,1,1,2,2,3,3,4,4,4\}$. For the circle, after one refinement, eight isogeometric elements will be produced, as shown in Fig. 2. Firstly, TL0 is the root including all boundary elements (eight isogeometric elements). Secondly, the eight isogeometric elements in TL0 are split into TL1-1 and TL1-2 which contain four isogeometric elements, respectively. Then, the TL1-1 and TL1-2 generated in second step are again separated into TL2-1 and TL2-2 as well as TL2-3 and TL2-4, respectively. This procedure is applied recursively until the number of tree levels reaches the user-given value R.

After constructing the binary tree, we will traverse it. Firstly, the whole boundary elements in the root (TL0) of the binary tree are separated into two parts. As shown in Fig. 3(a), since the elements are divided into two parts, four submatrices are generated, including two new off-diagonal submatrices (O11 and O12) and two new diagonal submatrices (D11 and D12). Then, the two off-diagonal submatrices (O11 and O12) will be approximated by the ACA algorithm, respectively. Diagonal submatrices do not need to be dealt with. Secondly, visit the next level (L = 1) of the binary tree. The boundary elements in the tree nodes (TL1-1 and TL1-2) are also separated into two parts, respectively. As shown in Fig. 3(b), the procedure for the binary tree (L = 1) will produce four new off-diagonal submatrices (O21, O22, O23 and O24) and four new diagonal submatrices (D21, D22, D23 and D24). As in the first step, the ACA algorithm is used to decompose the four off-diagonal submatrices. Repeat the above operation until $r = R - 1$. In this level, the boundary elements in each tree node are also divided into two parts, respectively. Different from the above process, the 2^r new diagonal matrices produced in this level should be evaluated directly. Up to now, the HODLR matrix has been constructed. Then, the inverse of the coefficient matrix is calculated by using the Sherman–Morrison–Woodbury formula.

As mentioned above, all the off-diagonal submatrices in the HODLR matrix are approximated by the ACA algorithm whose speed is related to the value of k in Eq. (19). When $k \ll \min(m, n)$, the storage space can be saved effectively, and the efficiency of the Sherman–Morrison–Woodbury formula can be improved substantially. However, if the ACA algorithm is used to decompose the off-diagonal submatrices such as O11 and O12 directly, the value k in Eq. (19) may be large and the implementation process may be slow, especially for large scale elastic problems [15]. In next section, we utilize an accelerated algorithm to decompose the off-diagonal submatrices into some small blocks and use the blocked scheme to accelerate the decomposition of off-diagonal submatrices.

4.1. The accelerated algorithm

According to the Ref. [24], considering an off-diagonal submatrix $\boldsymbol{B} \in \mathbb{R}^{m\times n}$, for the one-level accelerated algorithm, the matrix $\boldsymbol{B}$ is split into four parts given as (see Fig. 4, Level = 1)

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{13} & \boldsymbol{B}_{14} \end{bmatrix} \tag{23}$$

where $\boldsymbol{B}_{li} \in \mathbb{R}^{m_{li}\times n_{li}} (l = 1;\ i = 1, \ldots, 4)$. Then, the ACA algorithm is utilized to decompose the $\boldsymbol{B}_{li}$, thus $\boldsymbol{B}$ can be written as

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{U}_{11}\boldsymbol{V}_{11} & \boldsymbol{U}_{12}\boldsymbol{V}_{12} \\ \boldsymbol{U}_{13}\boldsymbol{V}_{13} & \boldsymbol{U}_{14}\boldsymbol{V}_{14} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_{11} & & \boldsymbol{U}_{12} & \\ & \boldsymbol{U}_{13} & & \boldsymbol{U}_{14} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_{11} & & \\ \boldsymbol{V}_{13} & & \\ & \boldsymbol{V}_{12} & \\ & \boldsymbol{V}_{14} & \end{bmatrix} \tag{24}$$

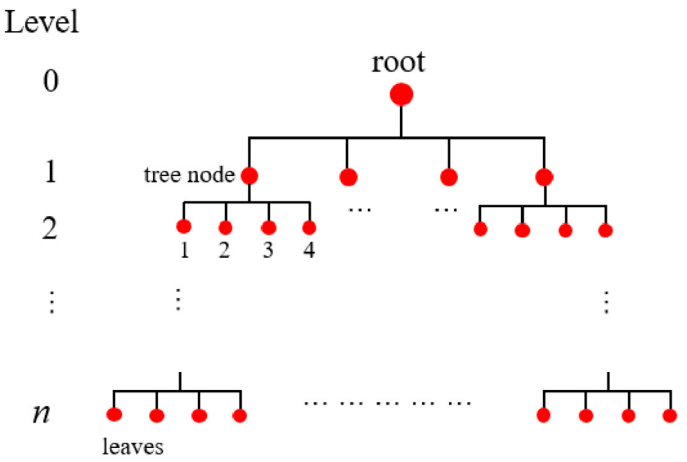


Fig. 5. Quadtree structure.

where $\boldsymbol{U}_{ij} \in \mathbb{R}^{m_{ij}\times k_{ij}}$ and $\boldsymbol{V}_{ij} \in \mathbb{R}^{k_{ij}\times n_{ij}}$ $(i = 1; j = 1, \ldots, 4)$. On the right side of Eq. (24), the dimension of first matrix is $m \times \sum_{j=1}^{4} k_{1j}$, and the dimension of second matrix is $\sum_{j=1}^{4} k_{1j} \times n$. The sum $\sum_{j=1}^{4} k_{1j}$ may be too large to use the Sherman–Morrison–Woodbury formula effectively [20]. Therefore, $\begin{bmatrix}\boldsymbol{V}_{11}\\ \boldsymbol{V}_{13}\end{bmatrix}$ and $\begin{bmatrix}\boldsymbol{V}_{12}\\ \boldsymbol{V}_{14}\end{bmatrix}$ need to be recompressed by ACA to obtain a satisfactory low-rank approximation as

$$\boldsymbol{B} = \boldsymbol{U}_1\boldsymbol{V}_1 \tag{25}$$

This idea can be implemented recursively. In the two-level scheme, we can deal with the submatrices $\boldsymbol{B}_{li}$ as the original matrix $\boldsymbol{B}$. The multi-level scheme and the corresponding algorithm can be found in [24].

4.2. The accelerated hybrid algorithm

The implementation of the accelerated algorithm described in Section 4.1 is shown in Fig. 4. As can be seen from Fig. 4, the algorithm adopts the quadtree structure as shown in Fig. 5. It is well known that the fundamental solutions shown in Eq. (2) are in tensor form, so the coefficient matrix of the elastic problems is not suitable for ACA decomposition [35]. Even if the ACA does not fail, the value k in Eq. (19) and the computation time will increase [15]. As mentioned in Section 3.1.1, for a given accuracy, the SVD can give a lowest rank. Hence, in the numerical implementation, the SVD is applied in the lowest level of the quadtree, namely the nth level in Fig. 5. The ACA is still used for the accelerated algorithm's recompression. The implementation of the accelerated hybrid algorithm is described as follows.

We need to build a quadtree. Firstly, one may choose any off-diagonal submatrix as the root node. Secondly, for simplicity, we divide the matrix into four blocks and use equal division, as shown in Fig. 4 (Level = 1). The four blocks are four tree nodes (Fig. 5 (Level = 1)), respectively. Then, the matrices 11, 12, 13 and 14 in Fig. 4 are also divided into four blocks, respectively, forming the tree nodes (Fig. 5 (Level = 2)). Divide recursively until the level reaches the given value n. The nodes which have no children are leaves.

After the quadtree is constructed, we will traverse it. Firstly, find the first leaf node, e.g. 3111 in Fig. 4, and decompose it with SVD. Then, other leaf nodes (3112, 3113, 3114) that share a common parent with 3111 are also decomposed by SVD. Secondly, use the recompression technique in Section 4.1 to form one low-rank matrix. This process continues until the Level = 0.

5. Postprocessing

5.1. Internal point stresses

The stress integral equation at the point $\boldsymbol{P}$ in the matrix is expressed as [6]

$$\sigma_{ij}(\boldsymbol{P}) = \sigma^0_{ij}(\boldsymbol{P}) + \int_\Gamma D_{ijk}(\boldsymbol{x}, \boldsymbol{P})t_k(\boldsymbol{x})d\Gamma(\boldsymbol{x}) - \int_\Gamma S_{ijk}(\boldsymbol{x}, \boldsymbol{P})u_k(\boldsymbol{x})d\Gamma(\boldsymbol{x}) \tag{26}$$

where D_{ijk} and S_{ijk} are the fundamental solutions of an isotropic elastic medium, which are defined as

$$D_{ijk} = \frac{1}{8\pi(1-v)r^2}\left[(1-2v)(\delta_{ik}r_{,j} + \delta_{jk}r_{,i} + \delta_{ij}r_{,k}) + 3r_{,i}r_{,j}r_{,k}\right] \tag{27a}$$

$$\begin{aligned} S_{ijk} = & \frac{G}{4\pi(1-v)r^3}\{3\frac{\partial r}{\partial n}\left[(1-2v)\delta_{ij}r_{,k}+v(\delta_{ik}r_{,j}+\delta_{jk}r_{,i})-5r_{,i}r_{,j}r_{,k}\right] \\ & +3v(r_{,i}r_{,k}n_j+r_{,j}r_{,k}n_i)+(1-2v)(\delta_{ik}n_j+\delta_{jk}n_i+3r_{,i}r_{,j}n_k) \\ & -(1-4v)\delta_{ij}n_k\} \end{aligned} \tag{27b}$$

For the Ith isotropic inclusion Ω_I, the corresponding stress integral equation at the point $\boldsymbol{P} \in \Omega^I$ is written as [6]

$$\sigma_{ij}^I(\boldsymbol{P}) = \int_{\Gamma_I} D_{ijk}^I(\boldsymbol{x},\boldsymbol{P})t_k^I(\boldsymbol{x})d\Gamma(\boldsymbol{x}) - \int_{\Gamma_I} S_{ijk}^I(\boldsymbol{x},\boldsymbol{P})u_k^I(\boldsymbol{x})d\Gamma(\boldsymbol{x}) \tag{28}$$

where $D_{ijk}^I = D_{ijk}$ and $S_{ijk}^I = \frac{G^I}{G}S_{ijk}$.

Based on Section 2, we can obtain the isogeometric discretization forms of Eqs. (26) and (28), respectively, as follows

$$\begin{aligned} \sigma_{ij}(\boldsymbol{P}) = & \sigma_{ij}^0(\boldsymbol{P}) + \sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}\left[\int_{-1}^{1}\int_{-1}^{1} D_{ijk}(\boldsymbol{P},\boldsymbol{x}(\hat{\xi},\hat{\eta}))R_J(\hat{\xi},\hat{\eta})J(\hat{\xi},\hat{\eta})d\hat{\xi}d\hat{\eta}\right]q_k \\ & - \sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}\left[\int_{-1}^{1}\int_{-1}^{1} S_{ijk}(\boldsymbol{P},\boldsymbol{x}(\hat{\xi},\hat{\eta}))R_J(\hat{\xi},\hat{\eta})J(\hat{\xi},\hat{\eta})d\hat{\xi}d\hat{\eta}\right]d_k \end{aligned} \tag{29}$$

and

$$\begin{aligned} \sigma_{ij}^I(\boldsymbol{P}) = & \sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}\left[\int_{-1}^{1}\int_{-1}^{1} D_{ijk}^I(\boldsymbol{P},\boldsymbol{x}(\hat{\xi},\hat{\eta}))R_J(\hat{\xi},\hat{\eta})J(\hat{\xi},\hat{\eta})d\hat{\xi}d\hat{\eta}\right]q_k^I \\ & - \sum_{e=1}^{NE}\sum_{J=1}^{(p+1)(q+1)}\left[\int_{-1}^{1}\int_{-1}^{1} S_{ijk}^I(\boldsymbol{P},\boldsymbol{x}(\hat{\xi},\hat{\eta}))R_J(\hat{\xi},\hat{\eta})J(\hat{\xi},\hat{\eta})d\hat{\xi}d\hat{\eta}\right]d_k^I \end{aligned} \tag{30}$$

Using Eqs. (29) and (30), one can obtain the internal point stress components in matrix and inclusions, respectively. However, when the computed point is very close to the inclusion–matrix interface, the nearly singular integrals will arise. The numerical results may be wrong if we do not deal with the nearly singular integrals of the integral equation. In this paper, we use the adaptive integration method [36] to deal with the nearly singular integrals in elastic problems, which has been used to calculate the nearly singular integrals in IGABEM [37].

5.2. Adaptive integral method

The standard Gauss quadrature formula in 3D can be given as [36]

$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)d\xi d\eta = \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} w_1^i w_2^j f(\xi^i,\eta^j) + E_1 + E_2 \tag{31}$$

where m_1 and m_2 are the number of Gauss points in the ξ and η directions, respectively. w_1^i and w_2^j are the corresponding weights. E_1 and E_2 are the integration errors in two directions ξ and η, respectively. The upper bound of relative error e_i in the ith direction is written as [38]

$$\frac{E_i}{I} \le 2\left(\frac{L_i}{4d}\right)^{2m_i}\frac{(2m_i+\beta-1)!}{(2m_i)!(\beta-1)!} \le e_i \tag{32}$$

where $\beta = 2$ is the order of integral singularity. The length of the element is L_i in the ith direction. d is the minimum distance between the source point and the element. The number of Gauss point m_i in the ith direction can be obtained as [36]

$$m_i = \sqrt{\frac{2}{3}\beta + \frac{2}{5}}[-0.1\ln(e_i/2)][(8L_i/3d)^{3/4}+1] \tag{33}$$

Then, rearrange Eq. (33) to yield

$$L_i = \frac{3}{8}d\left(\frac{-10m_i}{\sqrt{2q/3+2/4}\ln(e_i/2)} - 1\right)^{4/3} \tag{34}$$

For the adaptive integral method, in order to avoid using a large number of Gauss points, the element considered is repeatedly divided into sub-elements, thereby reducing the computational error. Through the adaptive integral method, the nearly singular integrals existing in IGABEM can be accurately computed by moderate Gauss orders. The details of derivation in IGABEM can be found in [37].

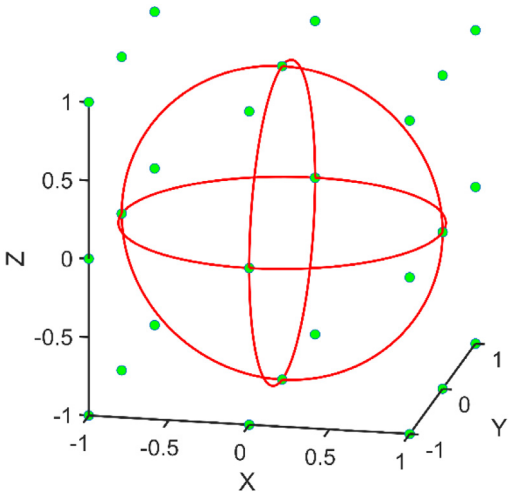


Fig. 6. The initial geometry for the spherical inclusion in which the control points are shown in the green dots.

6. Numerical examples

In this paper, the default tolerance ε_{ACA} and ε_{SVD} are set to be 10^{-5}, and the upper bound of relative error $e_i (i = 1, 2)$ in Eq. (32) is 10^{-6}. Both the SVD and inverse of the diagonal matrices are solved by Intel® Math Kernel Library (https://software.intel.com/en-us/mkl). To illustrate the efficiency and accuracy of the present algorithm, the following methods will be compared in this section.

- Method 1: The conventional IGABEM;
- Method 2: The HODLR scheme **without** the accelerated algorithm; and the approximation of off-diagonal submatrices **using** ACA for IGABEM;
- Method 3: The HODLR scheme **using** the accelerated algorithm; and the approximation of off-diagonal submatrices **using** ACA for IGABEM;
- Method 4: The HODLR scheme **using** the accelerated algorithm; and the approximation of off-diagonal submatrices **using** hybrid scheme for IGABEM.

In this section, there are four numerical examples with different models. In the examples, the polynomial degree of the NURBS is two, namely $p = 2$ and $q = 2$, unless otherwise specified. N denotes the degree of freedom. $E = 1$ is the Young's modulus of the matrix and E_I the Young's modulus of the inclusion. v represents the Poisson's ratio of the matrix which is equal to that of inclusions, and its value is 0.3. T_{total} represents the total CPU time of solving the system of linear equations, and T_{approx} is the CPU time of low-rank approximation for the upper right corner submatrix of HODLR matrix. k is the rank of low-rank matrix as mentioned in Section 3.1. The relative error in L_2 norm of the displacements and stresses with respect to the exact results is computed as

$$L_2 \text{ norm error } = \sqrt{\left\| \boldsymbol{f}_{num} - \boldsymbol{f}_{exact} \right\|^2 / \left\| \boldsymbol{f}_{exact} \right\|^2}$$

where $\boldsymbol{f}_{\text{num}}$ denotes the numerical result vector of the displacement or stress, and the $\boldsymbol{f}_{\text{exact}}$ is the corresponding exact result vector.

6.1. One spherical inclusion

One unit spherical inclusion embedded in an infinite domain is considered, as shown in Fig. 6, under the uniform remote tri-axial tension σ_0. Fig. 6 shows the initial elements described with the knot vectors $\Xi = \{0,0,0,0.25,0.25,0.5,0.5, 0.75,0.75,1,1,1\}$ and $\mathcal{H} = \{0,0,0,0.5,0.5,1,1,1\}$ as well as the polynomial orders $p = 2$ and $q = 2$. In this example, we investigate the effect of different Young's modulus ratios E_I/E on the stress distribution. The exact analytical solution can be found in [5].

For the sake of simplicity, we use the soft inclusion ($E_I/E = 0.5$) to test the accuracy and convergence of the present algorithm unless otherwise specified. For the convergence plot, the element size is calculated as $h = \sqrt{A^e_{\max}/A}$ in which $A^e_{\max}$ is the maximal area of the elements and A the surface area of Γ. Figs. 7 and 8 display the convergence curves in the L_2 normal error of the stresses and displacements, respectively, by four methods, i.e. Method 1, Method 2, Method 3

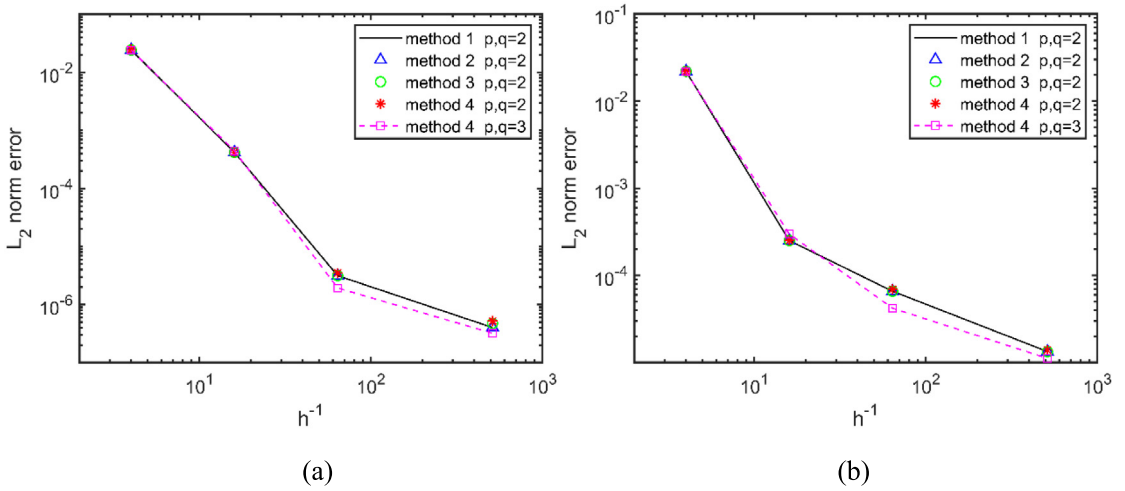


Fig. 7. The convergence of the stresses σ_{zz}: (a) Calculated points along $S1$ in the matrix part and (b) Calculated points along $S2$ in the inclusion part.

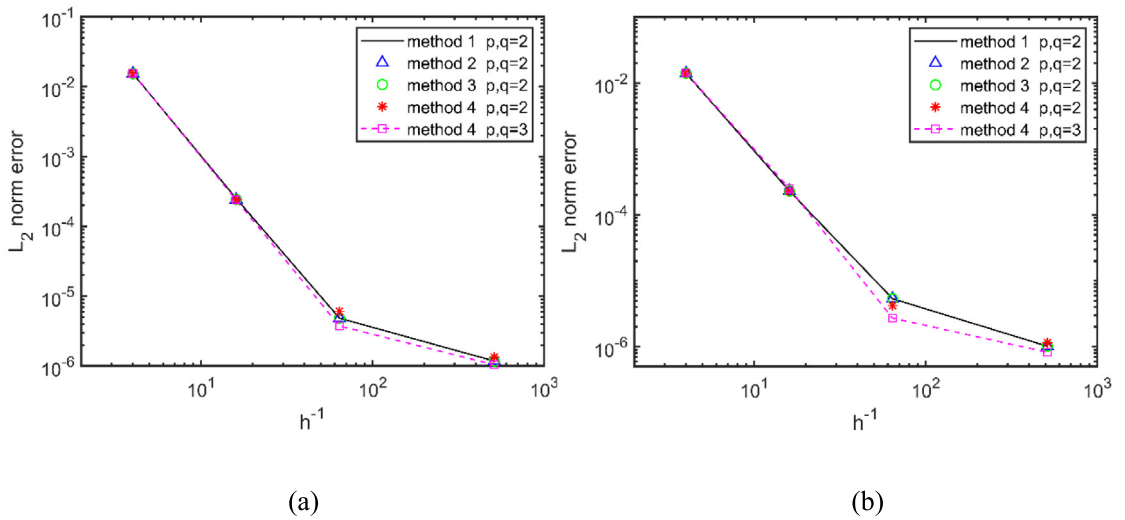


Fig. 8. The convergence of the displacements u_r: (a) Calculated points along $S1$ in the matrix part and (b) Calculated points along $S2$ in the inclusion part.

and Method 4. In Figs. 7 and 8, the calculated points in matrix part are distributed uniformly along the curve $S1$ and the calculated points in inclusion part are distributed uniformly along the curve $S2$, as follows

$$S1: x = R_{mat}\cos\theta,\ y = R_{mat}\sin\theta,\ z = 0;$$

$$S2: x = R_{inc}\cos\theta,\ y = R_{inc}\sin\theta,\ z = 0.$$

where $R_{mat} = 1.1$, $R_{inc} = 0.9$ and $\theta \in [0, 2\pi]$. From Figs. 7 and 8, it can be seen that the four methods are convergence and very close to each other. That is to say, we have improved the computing efficiency and kept the accuracy unchanged. With respect to method 4, both quadratic and cubic NURBS basis functions are investigated. By comparing the curves between quadratic and cubic NURBS basis functions in Figs. 7 and 8, it can be seen that when considering the higher-order NURBS basis functions, the L_2 normal errors are not improved. For this phenomenon, the problem may lie in the method of solving singular integrals. When the power series expansion method is used in 3D problems, a system of linear equations should be solved, in which condition number of the coefficient matrix is related to the parametric M ($M = 2^*(p + q)-2$). Therefore, when considering the higher-order NURBS basis functions, the parametric M will increase, resulting in larger condition number and no improvement in error. As the main purpose of this paper is to investigate the fast direct solver, this phenomenon will be discussed in depth in the further research. Table 1 shows the numerical results of the normalized stress σ_{zz}/σ_0 obtained by four methods. The exact results at the internal points along x axis are also given in

Table 1
The normalized stresses σ_{zz}/σ_0 along x axis with respect to N for four methods.

N	x	Method 1	Method 2	Method 3	Method 4	Exact
3456	0.5	0.72393	0.72393	0.72393	0.72393	0.72414
	0.99	0.79561	0.79561	0.79561	0.79562	0.72414
	0.999	17.2780	17.2780	17.2780	17.2781	0.72414
	1.001	31.4219	31.4219	31.4219	31.4218	1.13752
	1.01	0.98829	0.98829	0.98829	0.98829	1.13387
	1.1	1.10363	1.10363	1.10363	1.10363	1.10363
	3.0	1.00566	1.00566	1.00566	1.00566	1.00566
	5.0	1.00110	1.00110	1.00110	1.00110	1.00110
13 824	0.5	0.72405	0.72405	0.72405	0.72405	0.72414
	0.99	0.72297	0.72297	0.72297	0.72298	0.72414
	0.999	21.0260	21.0260	21.0260	21.0261	0.72414
	1.001	39.4940	39.4940	39.4940	39.4940	1.13752
	1.01	1.13659	1.13659	1.13660	1.13660	1.13387
	1.1	1.10363	1.10363	1.10363	1.10363	1.10363
	3.0	1.00566	1.00566	1.00566	1.00566	1.00566
	5.0	1.00110	1.00110	1.00110	1.00110	1.00110

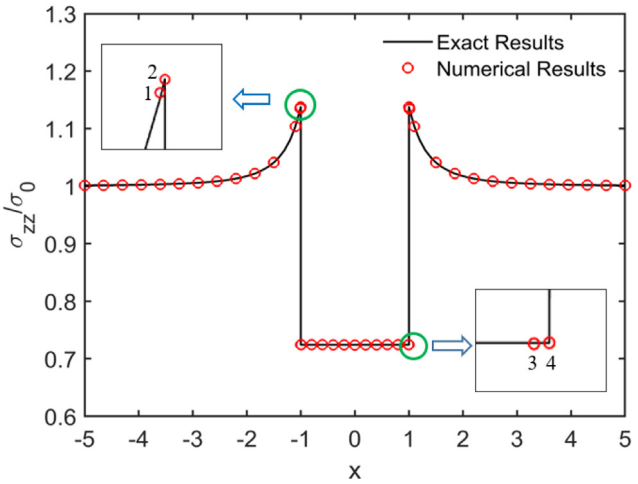


Fig. 9. The results of the normalized stress σ_{zz}/σ_0 along x axis for Method 4.

Table 1 for comparison. From Table 1, we can see that when the computed point is far from inclusion–matrix surface, the numerical results obtained by four methods are well consistent with the exact results. Even though the numerical results are convergence, with the computed point closer to the inclusion–matrix surface, nearly singular integrals will appear and the numerical results obtained by the standard Gauss quadrature method become unsatisfactory (such as $x = 0.999$ or $x = 1.001$). To evaluate nearly singular integrals, the adaptive integral method mentioned in Section 5 will be used in the program code. When $N = 3456$, the results of the normalized stress σ_{zz}/σ_0 obtained by Method 4 which uses the adaptive integral method and the exact results are displayed in Fig. 9, respectively. The numerical results at the calculated points (-1.01, -1.001, 0.99 and 0999) are shown in the small square frames (1, 2, 3 and 4 points) in Fig. 9, respectively. Fig. 10 shows the distribution of normalized stress σ_{zz}/σ_0 on the xy cross section by Method 4 for the soft inclusion and hard inclusion, respectively. From Fig. 10, we can see that the numerical results match the exact results very well. We also can find that for the soft inclusion the local normalized stress $\sigma_{zz}/\sigma_0$of matrix increases, whereas for the hard inclusion the local normalized stress σ_{zz}/σ_0 of matrix decreases. In order to see the error distribution more clearly, the relative error distribution of σ_{zz} for the soft inclusion on the xy cross section is plotted in Fig. 11 with Method 4. The red lines in the matrix and inclusion in Fig. 11 are curves $S1$ and $S2$, respectively. From Fig. 11, we can see that the larger relative error values appear near the inclusion–matrix interface and distribute between curves $S1$ and $S2$, which are computed by the adaptive integral method. Although the relative error value increases near the inclusion–matrix interface, the order of the magnitude of maximum relative error is 10^{-4}. Therefore, the present method can effectively deal with the elastic inclusion problems.

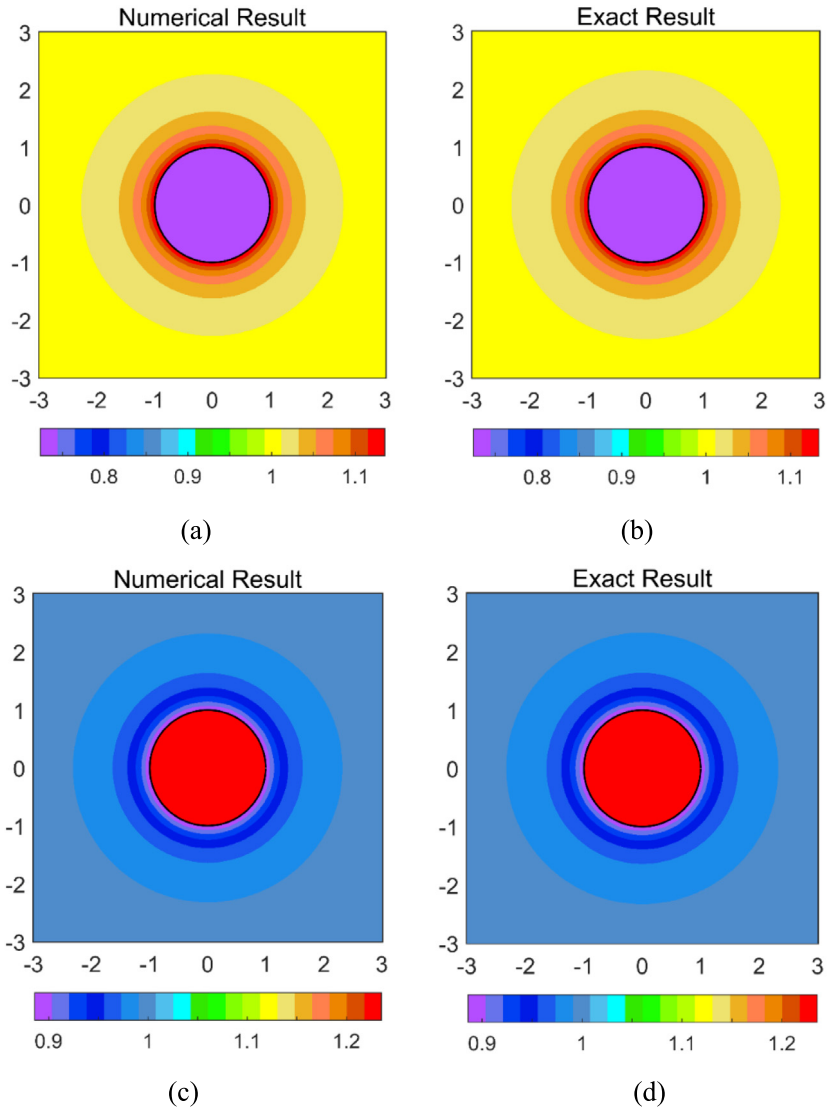


Fig. 10. The normalized stresses σ_{zz}/σ_0 on the xy cross section: (a) Numerical solution of one soft inclusion; (b) Exact solution of one soft inclusion; (c) Numerical solution of one hard inclusion; (d) Exact solution of one hard inclusion.

Table 2
The CPU time and rank k for four methods with respect to one spherical inclusion.

N		k	T_{approx} (s)	T_{total} (s)
3456	Method 1	–	–	247.91
	Method 2	472	81.084	170.22
	Method 3	727	35.009	78.249
	Method 4	407	4.4290	16.298
13 824	Method 1	–	–	14 627.1
	Method 2	1070	4236.0	8994.4
	Method 3	1326	1042.0	2221.1
	Method 4	713	153.60	436.62

In order to investigate the CPU time of solving the system of linear equations, Table 2 gives the CPU time of four methods for solving Eq. (13). As we can see from Table 2, Method 4 takes the least time and storage space of all the methods. Hence, for the same tolerance $\varepsilon_{ACA} = \varepsilon_{SVD} = 10^{-5}$, Method 4 requires the least memory and CPU time. As

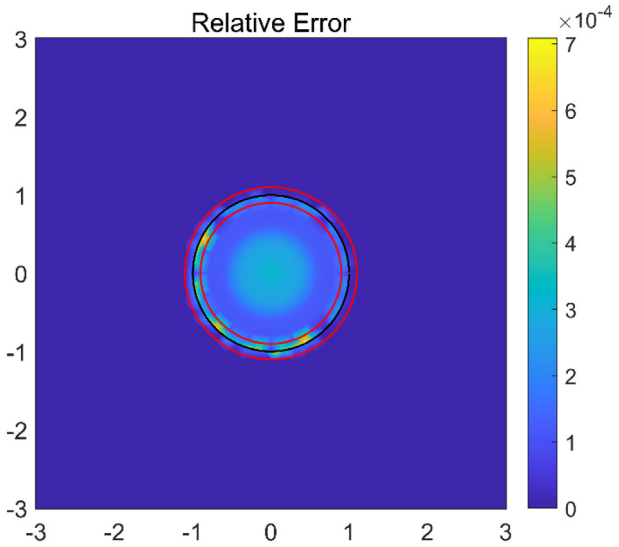
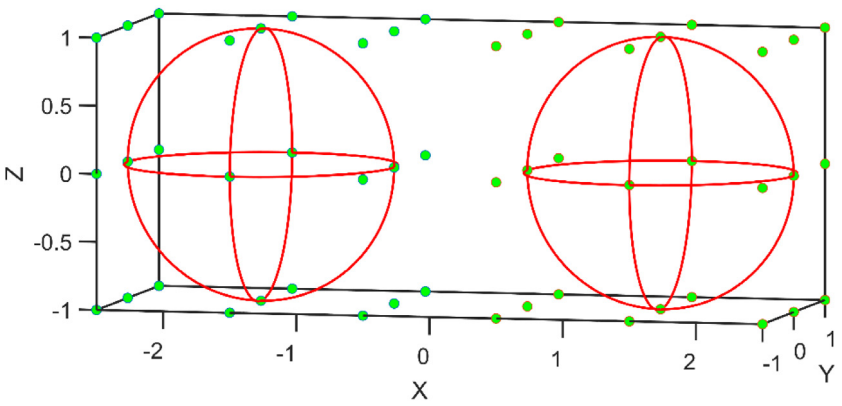
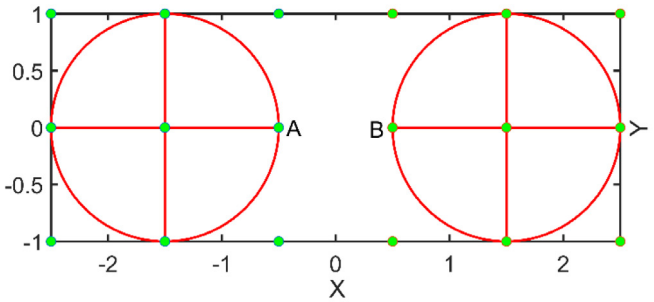


Fig. 11. The relative errors of the soft inclusion on the *xy* cross section.



(a)



(b)

Fig. 12. (a) The initial geometry for the two spherical inclusions in which the control points are shown in the green dots. (b) The vertical view of two spherical inclusions.

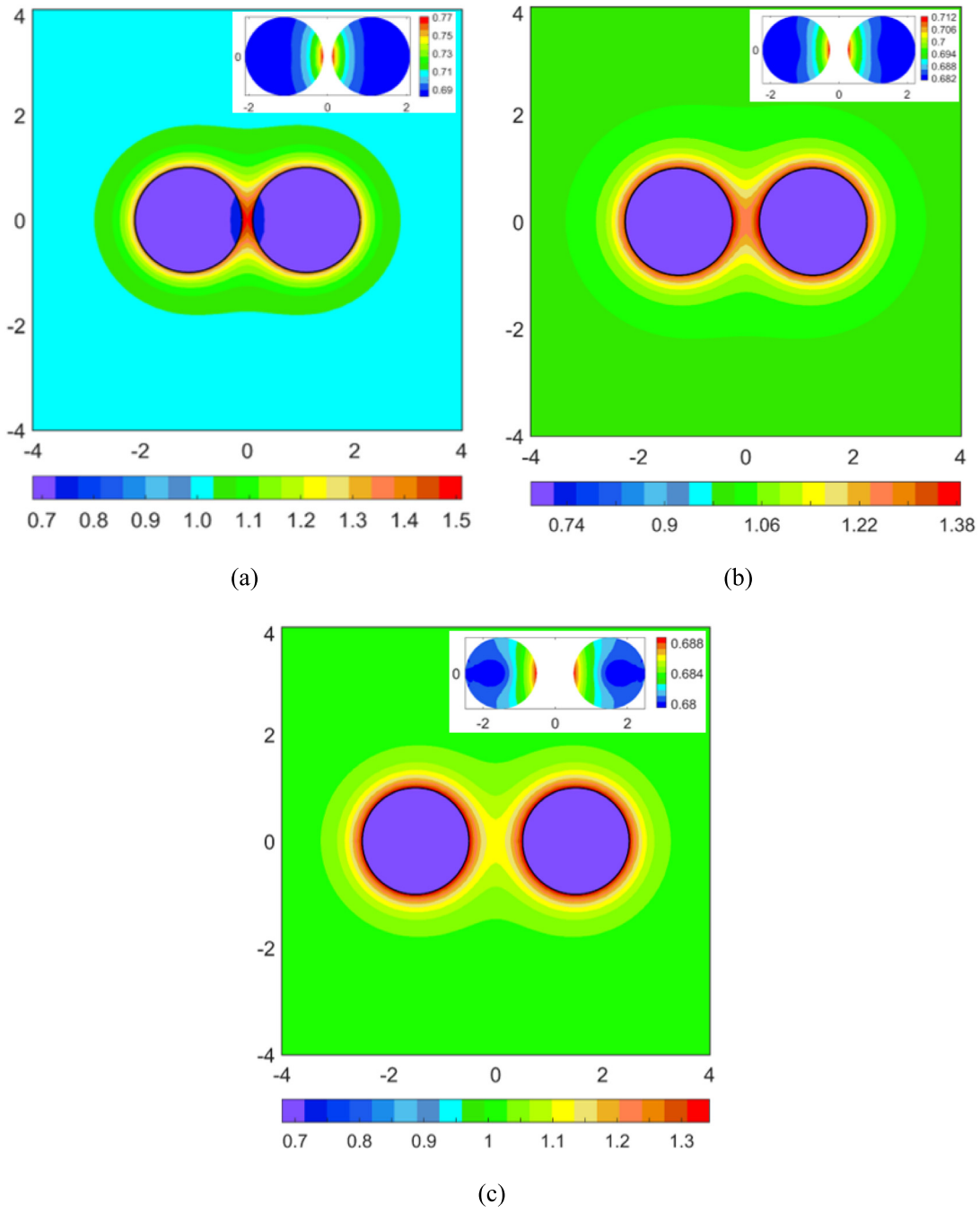


Fig. 13. The normalized stress σ_{zz}/σ_0 of two soft inclusions on the xy cross section: (a) Distance $d = 0.2$; (b) Distance $d = 0.5$; (c) Distance $d = 1$.

mentioned in Section 2, we need to solve two systems of linear equations for the considered problem. Therefore, with the increase of matrix size, the accelerated hybrid algorithm is more effective.

6.2. Two spherical inclusions

In the second example, we consider two unit spherical inclusions in an infinite matrix with the remote uniaxial loading $\sigma_0 = 1$ in z-direction, as shown in Fig. 12 which displays the initial elements and control points. $(-1.5,0,0)$ and $(1.5,0,0)$ are, respectively, the center coordinates of two spherical inclusions and $d = 1$ is the distance from A to B (see Fig. 12(b)).

In numerical implementation, 27 648 DOFs are applied to gain the numerical results. Fig. 13 shows the distribution of normalized stress σ_{zz}/σ_0 for soft inclusions on the xy cross section by Method 4. From Fig. 13, due to the existence

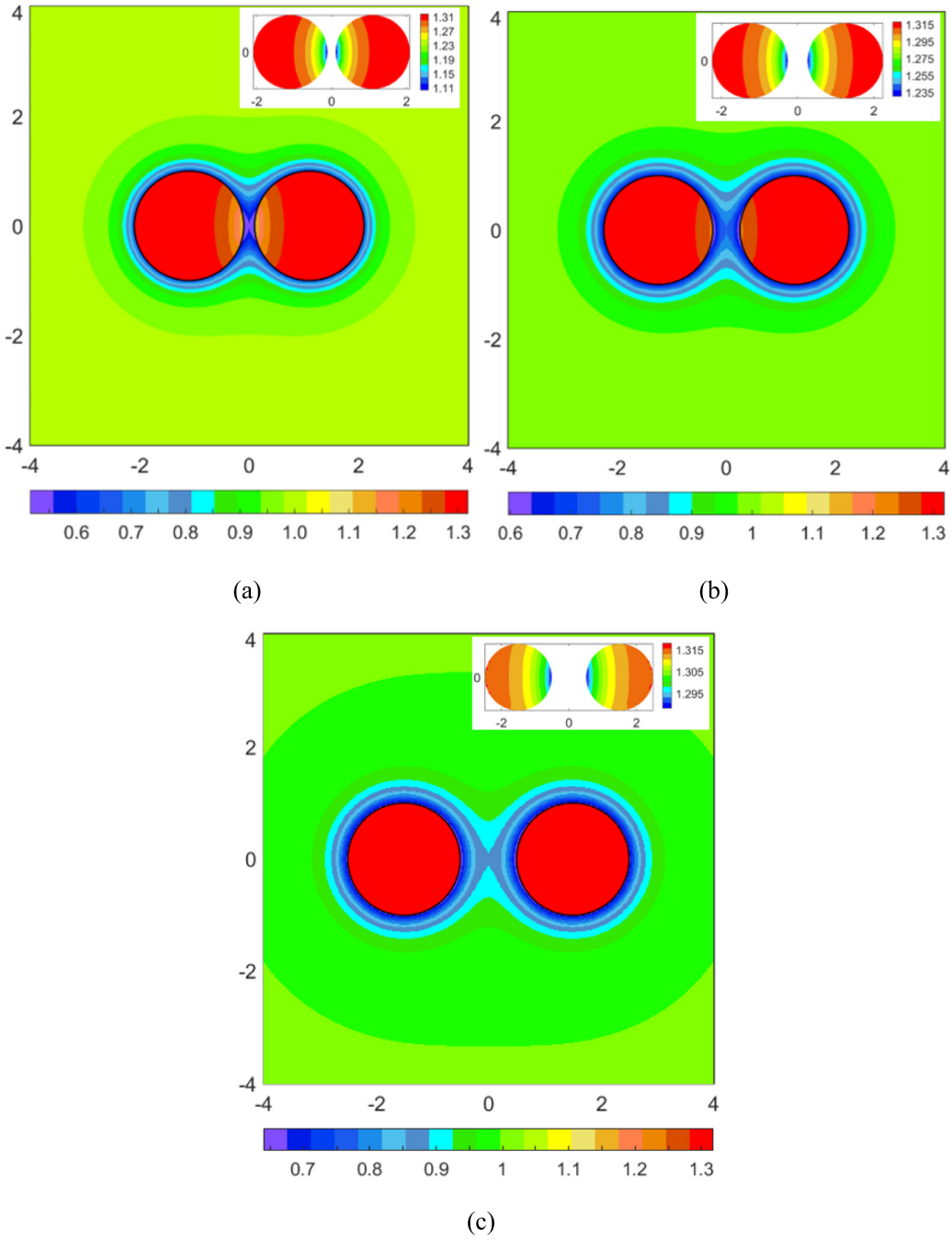


Fig. 14. The normalized stress σ_{zz}/σ_0 of two hard inclusions on the xy cross section: (a) Distance $d = 0.2$; (b) Distance $d = 0.5$; (c) Distance $d = 1$.

of the soft inclusions, we can clearly see that the local normalized stress σ_{zz}/σ_0 of matrix increases gradually near the inclusions and decreases from the boundary of inclusions to infinity until $\sigma_{zz}/\sigma_0 = 1$. In Fig. 13(a), (b) and (c), the distance d between two inclusions is 0.2, 0.5 and 1, respectively. In the upper right corner of Fig. 13(a), (b) and (c), the interaction of two inclusions is displayed, respectively. With the increase of the distance d between two inclusions, the interaction of two inclusions and the effect of two inclusions on the stress distribution become weak. When $d = 1$, we can find that the interaction of two inclusions almost disappeared, and the stress values vary between 0.68 and 0.688. Fig. 14 describes the normalized stress distribution σ_{zz}/σ_0 for hard inclusions in the same way as Fig. 13. Since the inclusions are hard, which is different from Fig. 13, the local normalized stress σ_{zz}/σ_0 of matrix in Fig. 14 decreases gradually near the inclusions and increases from the boundary of inclusions to infinity until $\sigma_{zz}/\sigma_0 = 1$. However, when the distance d between two

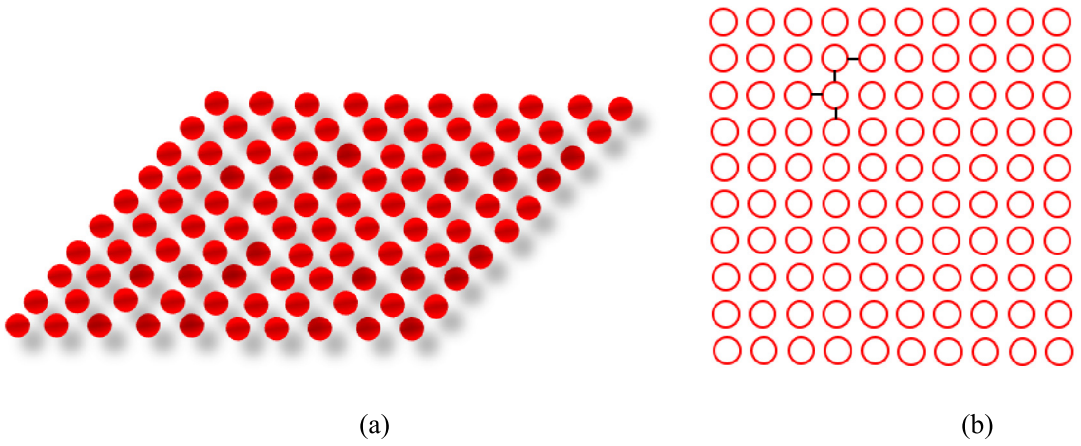


Fig. 15. (a) The geometric model of the 100 spherical inclusions in infinite space; (b) The vertical view of the 100 spherical inclusions in which the distance d between two spherical inclusions is represented by the black lines.

Table 3
The CPU time and rank k for three methods with respect to two spherical inclusions.

	Matrix approximation						Solving matrix		
	Method 2		Method 3		Method 4		Method 2	Method 3	Method 4
	T_{approx} (s)	k	T_{approx} (s)	k	T_{approx} (s)	k	T_{total} (s)	T_{total} (s)	T_{total} (s)
$d = 0.2$	21 569.2	1083	3070.7	2439	657.6	847	43 476.3	6575.4	1700.6
$d = 0.5$	11 514.9	761	1956.1	1742	474.6	587	23 357.3	4333.6	1404.5
$d = 1.0$	5868.1	492	1274.0	1222	399.1	420	12 139.2	2963.8	1206.6

Table 4
The CPU time and rank k for three methods with respect to 100 spherical inclusions.

	T_{approx} (s)	k	T_{total} (s)
Method 2	449 506.2	2662	928 311.4
Method 3	121 343.3	8102	251 683.2
Method 4	8986.0	2046	24 616.3

inclusions increases, the interaction of two inclusions and the effect of two inclusions on the stress distribution become weak, which is consistent with the case of soft inclusions.

Table 3 shows the CPU time and the rank k for solving Eq. (13) by three methods, i.e. Method 2, Method 3 and Method 4. The numerical computation is implemented on soft inclusions ($E_1/E = 0.5$). In Table 3, with the increase of distance d between two inclusions, the CPU time and storage space decrease for three methods. Method 4 uses the accelerated hybrid algorithm, which uses the least CPU time and memory space compared with Methods 2 and 3.

6.3. 100 spherical inclusions

In this example, 100 unit spherical inclusions are considered in an infinite matrix with the remote uniaxial loading $\sigma_0 = 1$ in z-direction, as shown in Fig. 15.

In Fig. 16, the distribution of normalized stress σ_{zz}/σ_0 for 100 soft inclusions on the xy cross section is described by Method 4. In order to investigate the effect of different distances on the distribution of stresses, the distances d between two inclusions are taken as 0.2, 0.5 and 1, respectively, as shown in Fig. 16(a), (b) and (c). As can be seen from Fig. 16, at different distances d, the stress distribution around the spherical inclusions is basically the same, but due to the soft inclusions, the stress around the spherical inclusions is relatively larger. The normalized stress σ_{zz}/σ_0 decreases from the boundary of inclusions to infinity until $\sigma_{zz}/\sigma_0 = 1$. However, as the distance d increases, the effect of the inclusions on the stress distribution becomes weak, and the interaction of inclusions also becomes weak. Compared with soft inclusions, Fig. 17 shows the distribution of normalized stress σ_{zz}/σ_0 for 100 hard inclusions. Different from the soft inclusions, as shown in Fig. 17, the normalized stress σ_{zz}/σ_0 of matrix becomes smaller around the hard inclusions, which increases from the boundary of inclusions to infinity until $\sigma_{zz}/\sigma_0 = 1$.

Numerical calculations are performed with 86,400 DOFs. The CPU time and the rank k for solving Eq. (13) are displayed in Table 4 by Methods 2, 3 and 4 on soft inclusion ($E_1/E = 0.5$). In Table 4, we only investigate one case where d is equal to 0.5. For large scale problems, CPU time and memory space in Method 4 are also minimal compared to Methods 2 and 3.

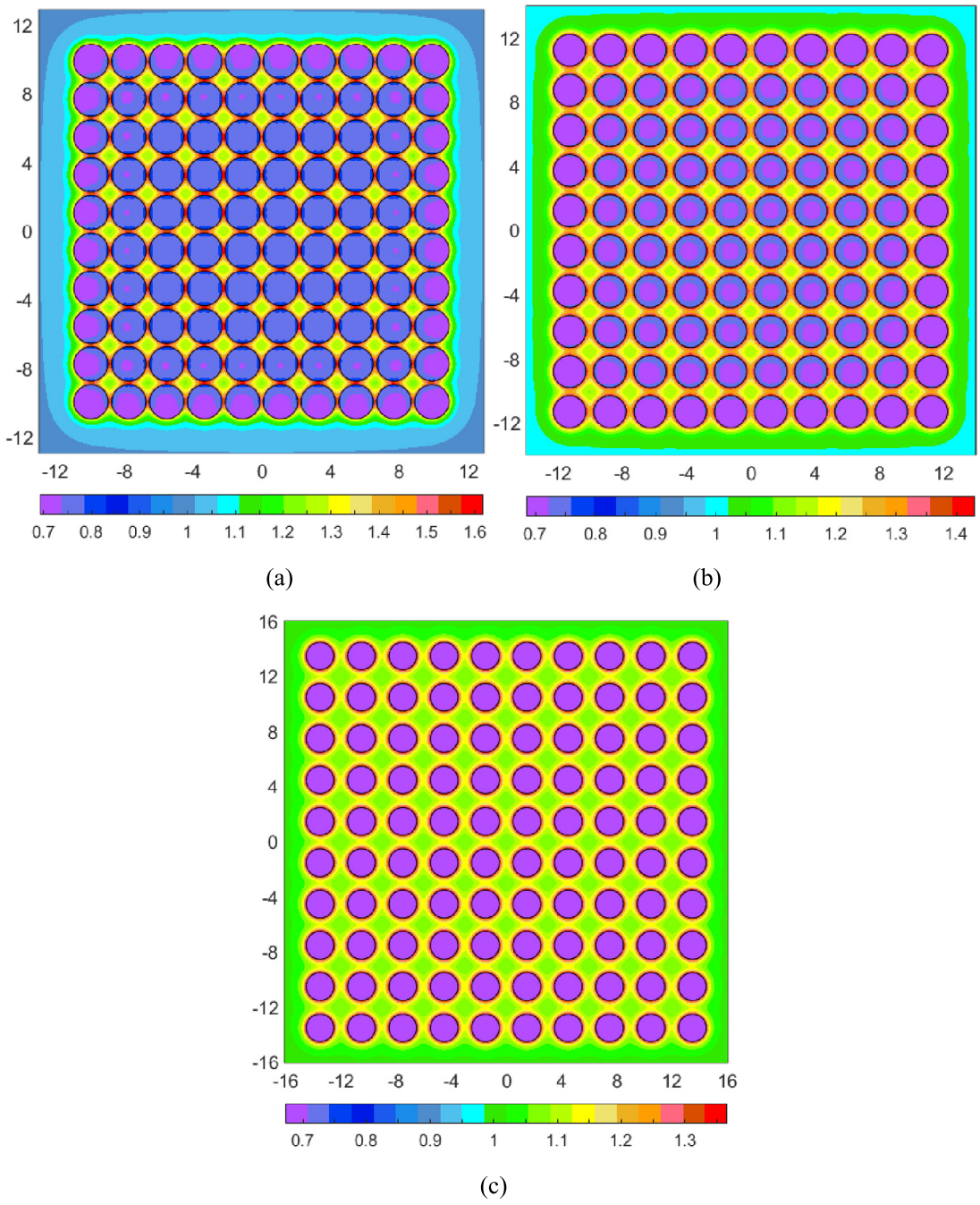


Fig. 16. The normalized stress σ_{zz}/σ_0 of 100 soft inclusions on the xy cross section: (a) Distance $d = 0.2$; (b) Distance $d = 0.5$; (c) Distance $d = 1$.

6.4. An inclusion of a complex shape

In the last example, an inclusion of complex shape in an infinite matrix subjected to the remote loading as in the example 6.3 is considered. The geometric model is showed in Fig. 18(a). $\Xi = \{0,0,0,0.25,0.25,0.5,0.5,0.75,0.75,1,1,1\}$ and $\mathcal{H} = \{0,0,0,1/6,1/6,1/3,1/3,1/2,1/2,2/3,2/3,5/6,5/6,1, 1,1\}$ are taken as the knot vectors in ξ and η directions and the polynomial orders are $p = 2$ and $q = 2$, respectively. Fig. 18(b) and (c) display the vertical view (xy cross section) and front view (xz cross section) of the inclusion, respectively, with the right view (yz cross section) identical to the front view.

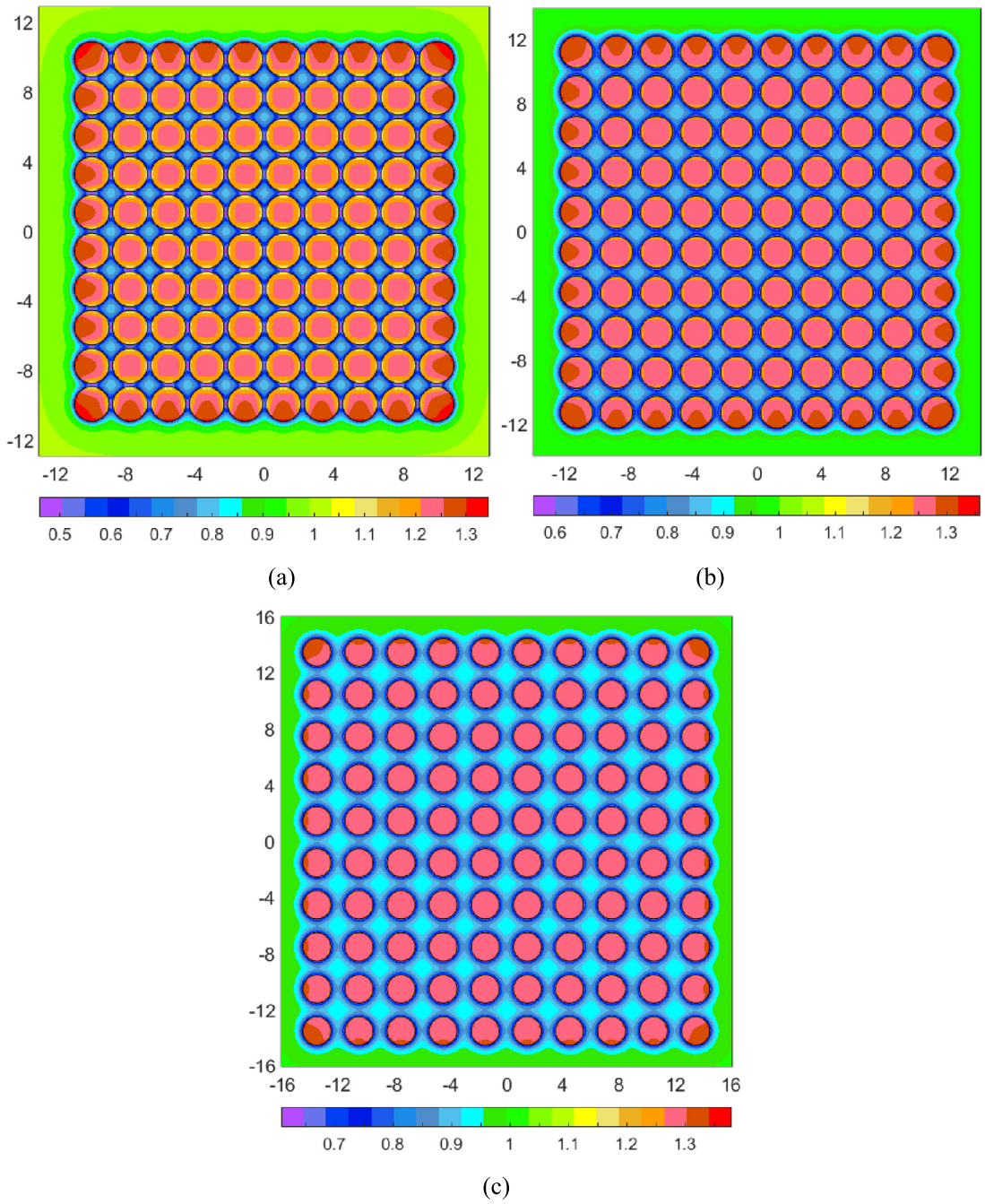


Fig. 17. The normalized stress σ_{zz}/σ_0 of 100 hard inclusions on the xy cross section: (a) Distance $d = 0.2$; (b) Distance $d = 0.5$; (c) Distance $d = 1$.

Fig. 19 presents the ABAQUS mesh of the inclusion with complex shape. As the reference solution, the calculated points are shown in Fig. 19(a) (xy cross section) and (b) (yz cross section), respectively. In Figs. 20 and 21, we investigate the Method 4 and use 10 368 DOFs to obtain the numerical results. Fig. 20 depicts the numerical results of normalized stress σ_{zz}/σ_0 at the calculated points with Method 4 and the FEM(ABAQUS) for soft inclusion ($E_I/E = 0.5$). The numerical results of xy and yz cross sections are shown in Fig. 20(a) and (b), respectively. Fig. 21 shows the numerical results of the normalized stress σ_{zz}/σ_0 for the hard inclusion ($E_I/E = 2$). Fig. 21(a) and (b) describe the numerical results on the xy and yz cross sections, respectively. From Figs. 20 and 21, it can be found that the numerical results of the present method are

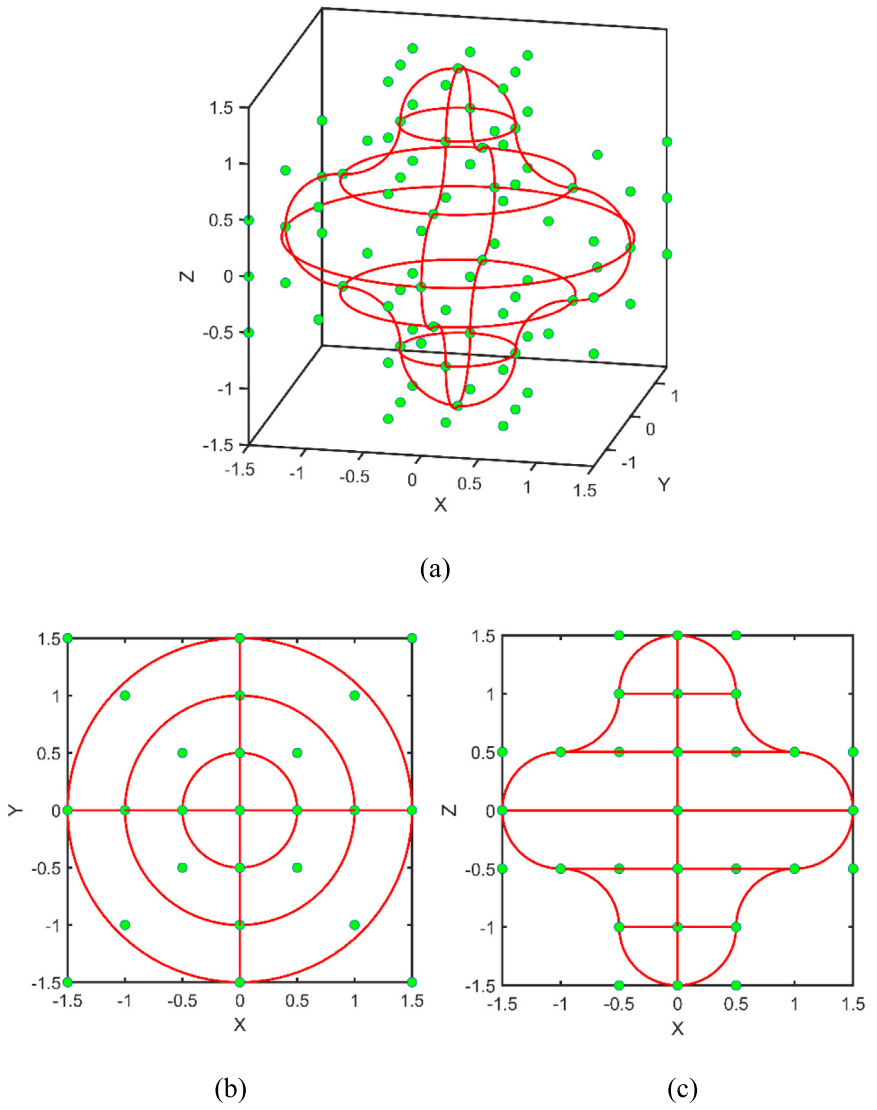


Fig. 18. (a) The initial geometry for the inclusion in which the control points are shown in the green dots. (b) The vertical view of the inclusion (xy cross section). (c) The front view of the inclusion (xz cross section).

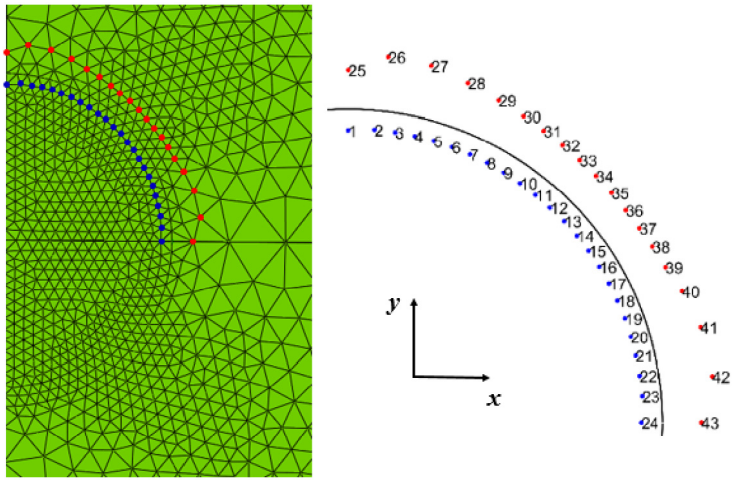
Table 5
The CPU time and rank k for four methods about the inclusion with complex shape.

	T_{approx} (s)	k	T_{total} (s)
Method 1	–	–	6807.9
Method 2	1158.0	1011	2583.9
Method 3	761.4	1353	1697.3
Method 4	99.8	645	302.7

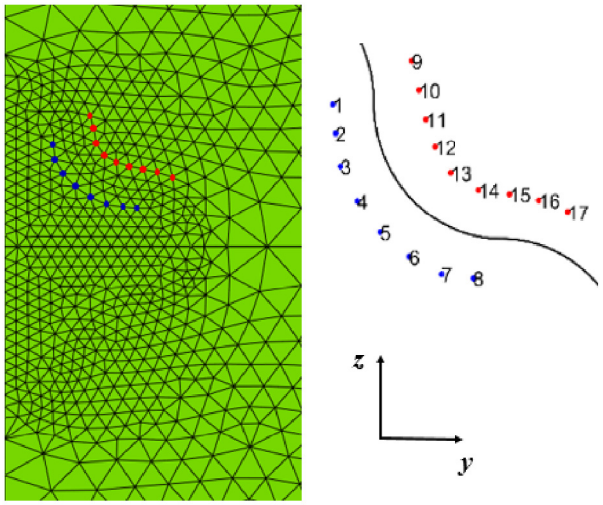
in agreement with the Abaqus results. Table 5 shows the CPU time and the rank k for solving Eq. (13) on soft inclusion ($E_{\text{I}}/E = 0.5$) by four methods.

7. Conclusions

Since the coefficient matrix of the IGABEM is dense and asymmetric, which limits the IGABEM to effectively solve large scale problems, this paper proposes a novel fast direct solver combining HODLR matrix and IGABEM to solve the 3D elastic inclusion problems. The concept of HODLR is that all off-diagonal submatrices are low-rank matrices. For large



(a)



(b)

Fig. 19. The ABAQUS mesh for the inclusion: (a) The *xy* cross section and (b) The *yz* cross section in which the red dots are the calculated points on the matrix and the blue dots are the calculated points on the inclusion. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

scale problems, we adopt the accelerated algorithm based on the divide-and-conquer method to avoid the direct use of ACA to decompose the off-diagonal submatrices. Thus, we can save the CPU time of ACA and reduce the value of rank for the obtained low-rank matrix. However, for the tensor fundamental solution of elastic problems, the SVD is utilized to approximate the bottom submatrices generated by the accelerated algorithm. In this paper, we have tried to study the interaction between multiple inclusions, including hard and soft inclusions. From the examples 2 and 3, we can find the distribution and variation of local stress. Numerical results also show that the proposed fast direct method can achieve satisfactory results with less CPU time and storage space.

Acknowledgments

The research is supported by the National Natural Science Foundation of China (11972085, 11672038).

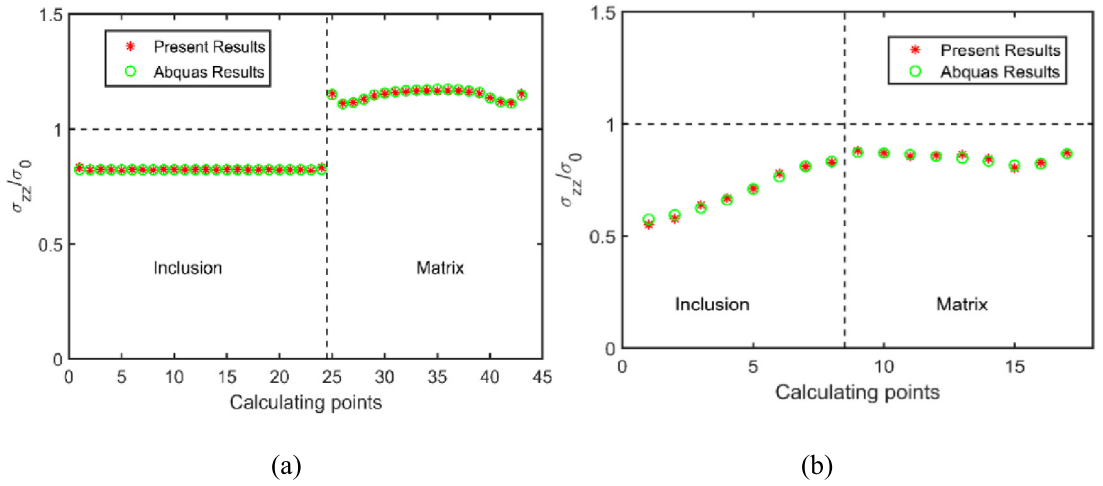


Fig. 20. The normalized stresses σ_{zz}/σ_0 of the calculated points for the soft inclusion: (a) The *xy* cross section and (b) The *yz* cross section.

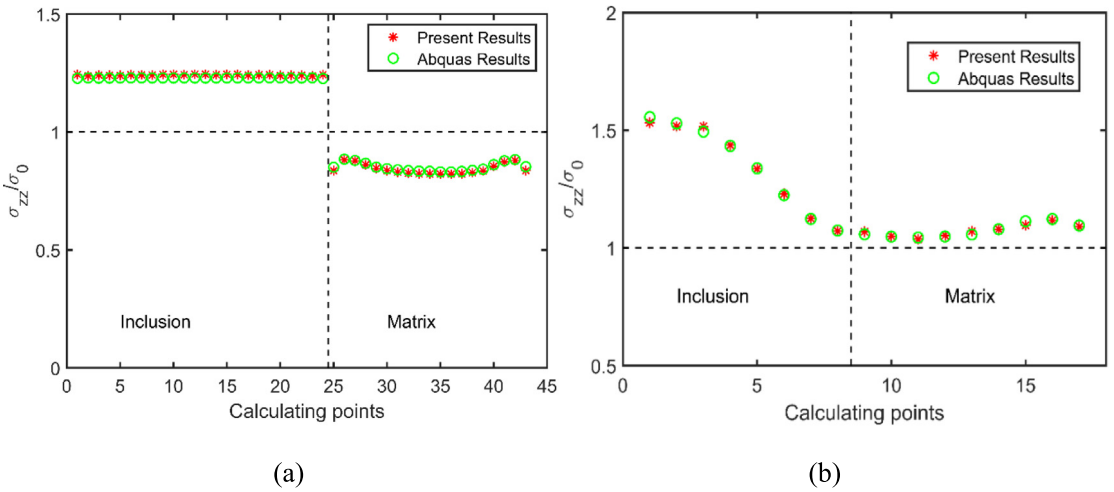


Fig. 21. The normalized stresses σ_{zz}/σ_0 of the calculated points for the hard inclusion: (a) The *xy* cross section and (b) The *yz* cross section.

References

[1] J.D. Eshelby, The determination of the elastic field of an ellipsoidal inclusion, and related problems, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 241 (1957) 376–396.
[2] N.I. Muskhelishvili, Some Basic Problems of Mathematics Theory of Elasticity, Noordhorff, Groningen, 1953.
[3] J. Zhang, N. Katsube, A hybrid finite element method for heterogeneous materials with randomly dispersed elastic inclusions, Finite Elem. Anal. Des. 19 (1–2) (1995) 45–55.
[4] T. Nakamura, S. Suresh, Effects of thermal residual stresses and fiber packing on deformation of metal-matrix composites, Acta Metall. Mater. 41 (6) (1993) 1665–1681.
[5] C.Y. Dong, S.H. Lo, Y.K. Cheung, Numerical solution of 3D elastostatic inclusion problems using the volume integral equation method, Comput. Methods Appl. Mech. Engrg. 192 (1–2) (2003) 95–106.
[6] C.Y. Dong, K.Y. Lee, A new integral equation formulation of two-dimensional inclusion–crack problems, Int. J. Solids Struct. 42 (18–19) (2005) 5010–5020.
[7] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39–41) (2005) 4135–4195.
[8] R.N. Simpson, S.P.A. Bordas, J. Trevelyan, T. Rabczuk, A two-dimensional isogeometric boundary element method for elastostatic analysis, Comput. Methods Appl. Mech. Engrg. 209–212 (2012) 87–100.
[9] A. Mantzaflaris, B. Jüttler, B.N. Khoromskij, U. Langer, Low rank tensor methods in Galerkin-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 316 (2017) 1062–1085.
[10] V. Rokhlin, Rapid solution of integral-equations of classical potential-theory, J. Comput. Phys. 60 (2) (1985) 187–207.
[11] R. Coifman, V. Rokhlin, S. Wandzura, The fast multipole method for the wave equation: A pedestrian prescription, IEEE Antennas Propag. Mag. 35 (3) (1993) 7–12.

[12] W. Hackbusch, Z. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering, Numer. Math. 54 (4) (1989) 463–491.
[13] S. Mallat, A theory for multiresolution signal decomposition: The wavelet representation, IEEE Trans. Pattern Anal. 11 (7) (1989) 674–693.
[14] W. Hackbusch, A sparse matrix arithmetic based on H-matrices, Computing 62 (1999) 89–108.
[15] B. Marussig, J. Zechner, G. Beer, T.P. Fries, Fast isogeometric boundary element method based on independent field approximation, Comput. Methods Appl. Mech. Engrg. 284 (2015) 458–488.
[16] W. Hackbusch, B.N. Khoromskij, A sparse H-matrix arithmetic, Computing 64 (2000) 21–47.
[17] M. Bebendorf, Approximation of boundary element matrices, Numer. Math. 86 (2000) 565–589.
[18] M. Bebendorf, S. Rjasanow, Adaptive low-rank approximation of collocation matrices, Computing 70 (2003) 1–24.
[19] J. Lai, S. Ambikasaran, L.F. Greengard, A fast direct solver for high frequency scattering from a large cavity in two dimensions, SIAM J. Sci. Comput. 36 (2014) 887–903, 2014.
[20] S. Huang, Y.J. Liu, A new fast direct solver for the boundary element method, Comput. Mech. 60 (2017) 1–14.
[21] W.K. Kong, J. Bremer, V. Rokhlin, An adaptive fast direct solver for boundary integral equations in two dimensions, Appl. Comput. Harmon. Anal. 31 (2011) 346–369.
[22] P.G. Martinsson, V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys. 205 (2005) 1–23.
[23] L. Greengard, D. Gueyffier, P.G. Martinsson, V. Rokhlin, Fast direct solvers for integral equations in complex three-dimensional domains, Acta Numer. 18 (2009) 1–23.
[24] F.L. Sun, C.Y. Dong, Y.H. Wu, Y.P. Gong, Fast direct isogeometric boundary element method for 3D potential problems based on HODLR matrix, Appl. Math. Comput. 359 (2019) 17–23.
[25] J. Sherman, W.J. Morrison, Adjustment of an inverse matrix corresponding to a change in one element of a given matrix, Ann. Math. Stat. 21 (1950) 124–127.
[26] X.W. Gao, An effective method for numerical evaluation of general 2D and 3D high order singular boundary integrals, Comput. Methods Appl. Mech. Engrg. 199 (2010) 2856–2864.
[27] A. Aimi, F. Calabrò, M. Diligenti, M.L. Sampoli, G. Sangalli, A. Sestini, Efficient assembly based on B-spline tailored quadrature rules for the IgA-SGBEM, Comput. Methods Appl. Mech. Engrg. 331 (2018) 327–342.
[28] F. Calabrò, A. Falini, M.L. Sampoli, A. Sestini, Efficient quadrature rules based on spline quasi-interpolation for application to IGA-BEMs, J. Comput. Appl. Math. 338 (2018) 153–167.
[29] R.R. Hiemstra, G. Sangalli, M. Tani, F. Calabrò, T.J.R. Hughes, Fast formation and assembly of finite element matrices with application to isogeometric linear elasticity, Comput. Methods Appl. Mech. Engrg. 355 (2019) 234–260.
[30] Y.P. Gong, C.Y. Dong, X.C. Qin, An isogeometric boundary element method for three dimensional potential problems, J. Comput. Appl. Math. 313 (2017) 454–468.
[31] C.A. Brebbia, J. Dominguez, Boundary Elements—An Introduction Course, Computational Mechanics Publications, New York, 1992.
[32] L.G.S. Leite, H.B. Coda, W.S. Venturini, Two-dimensional solids reinforced by thin bars using the boundary element method, Eng. Anal. Bound. Elem. 27 (2003) 193–201.
[33] F. Auricchio, L.B.D. Veiga, T.J.R. Hughes, A. Reali, G. Sangalli, Isogeometric collocation methods, Math. Models Methods Appl. Sci. 20 (11) (2010) 2075–2107.
[34] S. Voronin, P.G. Martinsson, RSVDPACK: An implementation of randomized algorithms for computing the singular value, interpolative, and CUR decompositions of matrices on multi-core and GPU architectures, 2016, arXiv:1502.05366v3.
[35] M. Bebendorf, Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems, Springer, 2008.
[36] X.W. Gao, T.G. Davies, Adaptive integration in elasto-plastic boundary element analysis, J. Chin. Inst. Eng. 23 (3) (2000) 349–356.
[37] Y.P. Gong, C.Y. Dong, An isogeometric boundary element method using adaptive integral method for 3D potential problems, J. Comput. Appl. Math. 319 (2017) 141–158.
[38] G.G.W. Mustoe, Advanced integration schemes over boundary elements and volume cells for two- and three-dimensional non-linear analysis, in: Developments in Boundary Element Methods-3, Elsevier, London, 1984, pp. 213–270.