Engineering Computations

Acceleration of free-vibrations analysis with the Dual Reciprocity BEM based on $\mathcal{H}$-matrices and CUDA

Yixiong Wei Qifu Wang Yunbao Huang Yingjun Wang Zhaohui Xia

Acceleration of free-vibrations analysis with the Dual Reciprocity BEM based on $\mathscr{H}$-matrices and CUDA

Yixiong Wei, Qifu Wang, Yunbao Huang, Yingjun Wang and Zhaohui Xia

National CAD Support Software Engineering Research Center, Huazhong University of Science and Technology, Wuhan, China

Received 2 July 2013
Revised 4 March 2014
11 April 2014
Accepted 22 April 2014

Abstract

Purpose – The purpose of this paper is to present a novel strategy used for acceleration of free-vibration analysis, in which the hierarchical matrices structure and Compute Unified Device Architecture (CUDA) platform is applied to improve the performance of the traditional dual reciprocity boundary element method (DRBEM).

Design/methodology/approach – The DRBEM is applied in forming integral equation to reduce complexity. In the procedure of optimization computation, $\mathscr{H}$-Matrices are introduced by applying adaptive cross-approximation method. At the same time, this paper proposes a high-efficiency parallel algorithm using CUDA and the counterpart of the serial effective algorithm in $\mathscr{H}$-Matrices for inverse arithmetic operation.

Findings – The analysis for free-vibration could achieve impressive time and space efficiency by introducing hierarchical matrices technique. Although the serial algorithm based on $\mathscr{H}$-Matrices could obtain fair performance for complex inversion operation, the CUDA parallel algorithm would further double the efficiency. Without much loss in accuracy according to the examination of the numerical example, the relative error appeared in approximation process can be fixed by increasing degrees of freedoms or introducing certain amount of internal points.

Originality/value – The paper proposes a novel effective strategy to improve computational efficiency and decrease memory consumption of free-vibration problems. $\mathscr{H}$-Matrices structure and parallel operation based on CUDA are introduced in traditional DRBEM.

Keywords Adaptive cross-approximation, Compute unified device architecture, Dual reciprocity boundary element method, $\mathscr{H}$-matrices

Paper type Research paper

1. Introduction

Elastic structures with arbitrary shape and time-dependent boundary conditions need to be analyzed by numerical methods. With the development of science, many numerical methods have been introduced for elastodynamics, such as finite difference method (FDM), finite element method (FEM), boundary element method (BEM) and so on. Because of the sparse combination matrix, FEM takes the dominant position in most commercial software for solving elastodynamic problems. However, those pre-processing operations, such as the selection of 3D element, quality control for mesh process, adjustment of small features and so on, occupy a large portion of computation resources. Those disadvantages lead to the obstacle of the integration of CAD\CAE procedure, which means the engineers need to transform model between CAD software and CAE software with complex treatment again and again (Park and Dang, 2010; Bazilevs *et al.*, 2010). This kind of transformation obviously is tedious and inconvenient

Engineering Computations: International Journal for Computer-Aided Engineering and Software
Vol. 32 No. 2, 2015
pp. 211-233

0264-4401
DOI 10.1108/EC-07-2013-0176

for engineers. Compared with FEM and FDM, the BEM, due to the advantages such as dimensionality reduction, high precision, is much more suitable for fast pretreatment, self-adaptive structural analysis in engineering software. That leads to the possibility for integrating engineering analysis process into CAD, as discussed by Wang (2009) and Wang *et al.* (2013) about their research in elastostaics. Meanwhile, boundary face method is an efficient solution method to the boundary integral equation and makes direct use of the B-rep data of a solid entity that available in all CAD packages (Zhang *et al.*, 2009; Qin *et al.*, 2010). In this paper, the efficiency of new strategy with BEM for free-vibration problems is discussed.

Rizzo introduced boundary integral element method into elastostatics first (Rizzo, 1967), which marks the beginning of systematic development of BEM for numerical problems. Banaugh and Goldsmith (1963) applied BEM into dynamics for steady plane elastodynamics, Tai and Shaw (1974) and De Mey (1976) utilized BEM for free-vibration problems. Many researchers, such as Niwa *et al.*, Beskos, Dominguez, Friedman, made effort for applying BEM into elastodynamics either time-domain or frequency-domain (Rizos and Karabalis, 1998; Michel, 1987; Banerjee *et al.*, 1986; Karabalis and Beskos, 1984, 1985 Dominguez and Roesset, 1978a, b). Nevertheless, the non-symmetric, fully populated matrix and low stability in numerical results of BEM would result in huge workload in computations and memory space within elastodynamic. This reason makes BEM rarely be used for solving 3D elastodynamic problems in engineering (Liu *et al.*, 2011). In traditional time-domain BEM field, the computational complexity will be $O(N_t N_s^2)$ while N_t is the time steps and $N_s = O(n^2)$ denotes discretization scale with respect to $O(n)$ observation points, as compared to only $O(N_t N_s)$ for FEM and fast multipole method (FMM).

The Dual Reciprocity Boundary Element Method (DRBEM) proposed by Nardini and Brebbia (1983) was concluded requiring less computation time than traditional BEM either in time-domain or frequency-domain by Chirino *et al.* (1994). The formulation in DRBEM makes use of static fundamental solutions to weight the dynamic equilibrium equation. It also uses an approach which first approximates the displacement field by a finite series radial basis functions (RBFs), and then apply the reciprocal theorem twice to convert the inertial volume integral to a surface. This method combines the advantages of dimensionality reduction and simplicity of elastostatic foundation solution. Ahmad and Banerjee (1986) proposed another particular integral BEM approach for conversion from domain to surface ones, which is proved to be mathematically equivalent to DRBEM. Due to the introduction of reciprocity twice, the final numerical equation involves matrix-matrix production (MMP) and matrix inversion operation, both of which are of $O(N^3)$ complexity.

Lots of methods have been developed to reduce time and space complexity for numerical solution, such as the mosaic-skeleton approximation (Tyrtyshnikov, 1996; Aparinov and Setukha, 2010), the FMM (Rokhlin, 1985; Wei *et al.*, 2012; Zhang and Tanaka, 2007), PWTD (Nishimura, 2002) and the panel clustering method (Hackbusch and Nowak, 1989). Mosaic-skeleton is devised by Hut and Piet (1986) for the treatment of the gravitational N-body problem, which is based on hierarchical subdivision of the space into Octree cells and approach mutual action between cells through recursive scheme. Based on multipole expansions for approximating kernel functions, Greengard and Rokhlin (1997) presented a similar approach for particles problems. Referring to those early works, FMMs have been developed to solve boundary element formulations for different kinds of problems, Wei *et al.* (2012) has involved FMM for dealing with 3D elastodynamic simulation

procedure, but the efficiency is proved to be negative for complicated MMP operation. And panel clustering method requires the knowledge of some kernel expansion in advance to perform integration, and have scarcely effect referring to $O(N^3)$ matrix arithmetic operations.

This paper introduces another effective numerical method presented by Hackbusch (1999) with hierarchical tree structure, which is similar to mosaic-skeleton method in matrix compression consideration. Representing the coefficient matrix in hierarchical format, the solution of the system can be obtained either directly by inverting the matrix in hierarchical format, or indirectly by using iterative schemes with or without pre-conditioners (Grasedyck, 2005; Bebendorf, 2005). Tyrtyshnikov (1996) has observed that low-rank block approximations could be built from only a few entries of the original block. Then Bebendorf (2000) proposed the method for construction low-rank approximation through regularization evaluation with selected rows and columns (pivot in original block). Three years later, Bebendorf and Rjasanow (2003) have further developed this method and referred as adaptive cross-approximation (ACA). This algorithm has been demonstrated almost linear complexity either in computation efficiency or storage requirement. Recently, some other researchers have focussed on the optimization of ACA, such as Zhang *et al.* (2013). Considering accuracy and stability, ACA algorithm is considered for low-rank block approximation in this paper.

Hierarchical matrices and their arithmetic have been extensively studied, and their application has been proved successful for the analysis of some realistic problems. For example, Kurz *et al.* (2002) and Ostrowski *et al.* (2006) proposed applications for electromagnetic problems, and Bebendorf and Grzhibovskis (2006) proposed the application for elastic problems. Recently, more and more researchers took efforts to hierarchical matrices in elastodynamic field. Such as Chaillat's work in model seismic wave propagation and amplification with multi-domain BEM (Chaillat *et al.*, 2009); Benedetti' s work (Benedetti and Aliabadi, 2010) of involving hierarchical matrices into elastodynamic crack problems; and the work about anisotropic time-harmonic in 3D elastodynamics discussed by Milazzo *et al.* (2012).

During the evaluation numerical equation in DRBEM, the inversion arithmetic is necessarily to be noticed for high complexity in computation. In this paper, a parallel inversion algorithm applying Compute Unified Device Architecture (CUDA) technology is presented in Section 4, and speedup ratio is compared with $\mathscr{H}$-Matrices inversion algorithm. CUDA is proposed by NVIDIA Corporation in 2003 for accelerating general computation processes with graphics processing units (GPU). Because of the purely aim for calculation, this technique is suitable for numerical computation. At present, there are already some researchers taking CUDA into numerical problems, such as, Gumerov and Duraiswami (2008) pioneered fine-grained parallel FMM algorithm based on CUDA, Wang *et al.* (2013) takes advantage of CUDA to accelerate classical fast multipole BEM for elastatics. It is worth noting that the parallel algorithm has the limitation of the scale of solvable problem for limitation of the device memory space. There are two strategies for applying $\mathscr{H}$-Matrices with DRBEM to solve free-vibration problems in Section 4.2: one is covering all matrices referred in numerical equation with hierarchical structure for large-scale problems named purely $\mathscr{H}$-Matrices&DRBEM method (PHDM), and another strategy is only covering part of matrices to avoid complicated truncation operation for pursuing efficiency named mixed $\mathscr{H}$-Matrices&DRBEM method (MHDM).

This paper is organized as follows. First, the basic equations for free-vibration analysis through DRBEM are briefly reviewed. The hierarchical matrix structure and algorithm is discussed in detail in Section 3, and the serial and parallel algorithm for inverse operation are also presented in this part. Then, two different strategies with $\mathscr{H}$-Matrices&DRBEM are presented in Section 4. In the last section, some free-vibration problem models are discussed to address the effect of strategies presented in this paper.

2. DRBEM in elastodynamics

The computation process can be greatly simplified by using static fundamental functions instead of dynamic ones. But the inertial item will appear in the equation such that the domain field needs discretization, which makes the technique lose the attraction of its "boundary only" character. The DRBEM is essentially a generalized way of constructing particular solutions that can be used to represent internal source distribution. For the homogenous medium, the motion of linearly elastic body of volume Ω and surface S is described by Navier-Cauchy partial differential equation. Assuming zero body forces and initial conditions, as applying traditional BEM, one can obtain an integral representation.

The fundamental solution is very complicated for computation. Because of the time variables in integral equation, the complexity will be $O(N_t N_s^2)$ where N_t is the time steps and $N_s = O(n^2)$ denotes discretization scale with respect to $O(n)$ observation points, while the complexity is only $O(N_t N_s)$ for FEM and FMM. For simplifying the computation process, one can substitute static functional solutions with dynamic ones, such as:

$$\boldsymbol{c}(x)\boldsymbol{u}(x,t) = \int_{\Gamma} \left[\boldsymbol{u}^*(x,\xi)\boldsymbol{p}(\xi,t) - \boldsymbol{p}^*(x,\xi)\boldsymbol{u}(\xi,t)\right] d\Gamma(\xi) - \int_{\Omega} \boldsymbol{u}^*(x,\xi)\rho\ddot{\boldsymbol{u}}(\xi,t)d\Omega(\xi), \tag{1}$$

in which $\boldsymbol{u}^*(x,\xi)$ and $\boldsymbol{p}^*(x,\xi)$ are static fundamental solutions and the overdot indicate differentiation with respect to time. However, the appearance of the inertial volume integral in Equation (1) indicates the discretization in volume domain is unavoidable, which would eliminate the biggest advantage of BEM. Nardini and Brebbia (1983) transferred this volume integral to the boundary surface, thereby creating an all-boundary integral formulation that leads to DRBEM.

The key point of DRBEM is expressing the unknown $\boldsymbol{u}(\boldsymbol{x},t)$ in Ω as a series of production by unknown time-dependent coefficients $\alpha_i^m(t)$ and known basis function $f^m(x)$:

$$u_i(x,t) = \sum_{m=1}^{M} \alpha_i^m(t) f^m(x), \quad x \in \Omega, \tag{2}$$

in which $M = N+L$, and N and L are the number of boundary and internal collocation points, respectively. It is worth noting that L could be zero. According to the work of Agnantiaris *et al.* (2001) for 3D elastodynamics, the effect of augmentation in the linear RBF for accuracy is very small, and for non-axisymmetric 3D structures polynomial $1+r$ is simplest and high-accuracy RBF.

Using the reciprocity theorem again, one succeeds in transforming Equation (1) into a boundary integral form:

$$-\int_{\Omega} \boldsymbol{u}^{*}(\boldsymbol{x},\xi)\rho\,\ddot{\boldsymbol{u}}(\xi,t)d\boldsymbol{\Omega}(\xi)$$

$$=\rho\sum_{m=1}^{M}\ddot{\alpha}_{n}^{m}\left[c_{ij}(\boldsymbol{x})\kappa_{jn}^{m}(\boldsymbol{x})+\int_{\Gamma}p_{ij}^{*}(\boldsymbol{x},\xi)\kappa_{jn}^{m}(\boldsymbol{x})d\boldsymbol{\Gamma}(\xi)-\int_{\Gamma}u_{ij}^{*}(\boldsymbol{x},\xi)\zeta_{jn}^{m}(\boldsymbol{x})d\boldsymbol{\Gamma}(\xi)\right], \quad (3)$$

in which $\kappa_{jn}^{m}(\boldsymbol{x})$ and $\zeta_{jn}^{m}(\boldsymbol{x})$ are the particular solutions of displacement and traction field, which corresponds to the function f introduced in Equation (2) for approximating the dynamic displacement field. The equation of these two solutions are listed in Appendix, respectively. Then, with the discretization of the boundary Γ into numbers of triangle elements (the total number of node is N), Equation (1) could form the matrix equation:

$$[\boldsymbol{M}]\{\ddot{\boldsymbol{u}}\}+[\boldsymbol{H}]\{\boldsymbol{u}\}=[\boldsymbol{G}]\{\boldsymbol{p}\}, \quad (4)$$

where:

$$[\boldsymbol{M}]=\rho([\boldsymbol{G}][\boldsymbol{P}]-[\boldsymbol{H}][\boldsymbol{W}])[\boldsymbol{F}]^{-1}, \quad (5)$$

in which $[\boldsymbol{H}]$ and $[\boldsymbol{G}]$ are the integral coefficient $N\times3\times N\times3$ matrices, $[\boldsymbol{F}]$ is the matrix contains the value of basis radial function f, and $[\boldsymbol{P}]$ and $[\boldsymbol{W}]$ are matrices containing sub-matrices of particular solution $\boldsymbol{\kappa}_{j}^{m}$ and $\boldsymbol{\zeta}_{j}^{m}$. When introducing the collocation points to improve accuracy, the matrices could be wrote as:

$$[\boldsymbol{M}]=\rho\left(\begin{bmatrix}\boldsymbol{G}_{BB}\\ \boldsymbol{G}_{DB}\end{bmatrix}\begin{bmatrix}\boldsymbol{P}_{BB} & \boldsymbol{P}_{BD}\end{bmatrix}-\begin{bmatrix}\boldsymbol{H}_{BB} & \boldsymbol{H}_{BD}\\ \boldsymbol{H}_{DB} & \boldsymbol{H}_{DD}\end{bmatrix}\begin{bmatrix}\boldsymbol{W}_{BB} & \boldsymbol{W}_{BD}\\ \boldsymbol{W}_{DB} & 0\end{bmatrix}\right)[\boldsymbol{F}]^{-1}, \quad (6)$$

in which the subscript $\boldsymbol{B}$ and $\boldsymbol{D}$ represent the boundary nodes and interior collocation points, respectively.

In modal analysis, considering time-harmonic dependence for the boundary displacement and traction vectors appearing in Equation (4). The frequency-domain equation is:

$$\left(-\omega^{2}[\boldsymbol{M}]+[\boldsymbol{H}]\right)\{\boldsymbol{u}_{0}\}=[\boldsymbol{G}]\left\{\boldsymbol{p}_{0}\right\}, \quad (7)$$

in which ω is the circular frequency of the harmonic excitation of $\boldsymbol{u}$ and $\boldsymbol{p}$ vectors with amplitude $\boldsymbol{u}_{0}$ and $\boldsymbol{p}_{0}$, respectively. Just setting the external disturbances equal to zero, one can obtain numerical equation of free vibration problems:

$$[\boldsymbol{A}]\{\boldsymbol{x}\}=\omega^{2}\left[\boldsymbol{M}^{*}\right]\{\boldsymbol{x}\}, \quad (8)$$

in which $[\boldsymbol{A}]$ is the BEM influence matrix referring to all unknown boundary variables contained in $\{x\}$, and $[\boldsymbol{M}^{*}]$ is obtained by setting zeros in $[\boldsymbol{M}]$ in which sub-columns refer to specified displacements.

3. Hierarchical matrices

3.1 Basic principles for $\mathscr{H}$-matrices

There are two kinds of representation to store matrix data, one is entrywise format and the other is out-product format (see Equation (9)):

$$A = \sum_{i=1}^{k} u_i v_i^H \tag{9}$$

where u_i and v_i is the ith vector, $i = 0,1, \ldots, k$, $A \in \mathbb{C}_k^{m \times n}$. Instead of storing $m \times n$ entries, we can store the matrix A with $m \times k$ matrix U and $n \times k$ matrix V, which requires $k(m+n)$ units of storage space. Hence, if the rank of k is small enough, it will save large memory space for storing matrix data. Hackbusch (1999) pioneered the $\mathscr{H}$-matrix format and their arithmetic: hierarchical matrices can be thought as a set of blocks connected in hierarchical tree structure, some of which satisfied admissible condition (Equation (12)) could be approximated by low-rank block for compression space, then the other leaf blocks are represented entirely.

Before obtaining the block, the partition of the matrix indexes through subdivision of nodes need to be executed, which leads to the so-called cluster tree. The cluster tree is used to form pairs of clusters recursively to define the block tree which provides the hierarchical block subdivision of the matrix. Applying recursive algorithm, the root collocation of nodes will be divided into binary subsets when the number of nodes in each cluster does not exceed the minimum *blocksize* n_{min}. However, different from FMM or other recursive algorithm with tree-boxes structure, those clusters are obtained from judging the rearrange index of node $z_i \in X_i$ by the plane (line for 2-dimension (2D)) whose normal direction is the main direction of father cluster. As seen from **A** of Figure 1, the symbol m_t is the centroid of t cluster, S is the judgment plane (judgment line in 2D), $\mu(X_i)$ is the distance from arbitrary point X_i to fixed point Z_i:

$$\sum_{i \in t} \left| w^T (z_i - m_t) \right|^2 = \max_{||v||_2} \sum_{i \in t} \left| v^T (z_i - m_t) \right|^2 \tag{10}$$

$$m_t = \frac{\sum_{i \in t} \mu(X_i) Z_i}{\sum_{i \in t} \mu(X_i)}$$

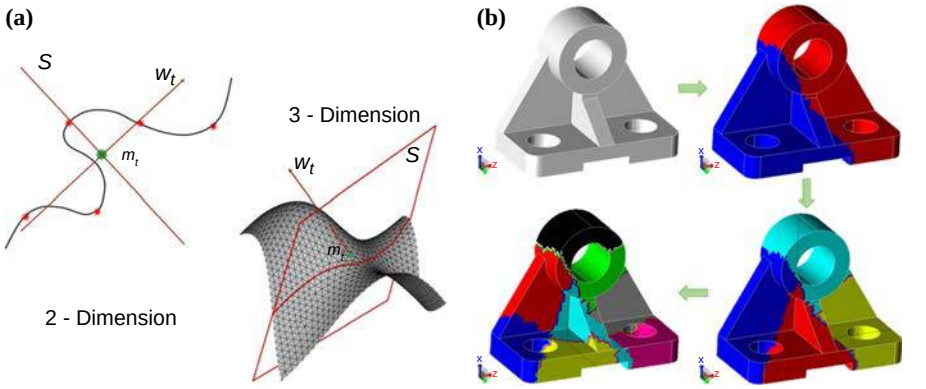

Figure 1.
Judge the position of nodes with main direction

And if w_t satisfies the equation in Equation (10), we will name it the main direction of surfaces in t cluster and that will also be the normal direction of S. We should know that w_t is actually attained as the eigenvector corresponding to the largest eigenvalue of C_t refer to Equation (11) (Bebendorf, 2008b). Through this kind of subdivision, balanced partition could be obtained, as shown in the example in **B** of Figure 1.

For symmetric positive semi-definite covariance matrix:

$$C_t = \sum_{i \in t} (z_i - m_t)(z_i - m_t)^T \in R^{d \times d} \tag{11}$$

During the partition process, the rearrangement of node index should be noticed before constructing $\mathscr{H}$-Matrices structure. The bijective map should be founded for tracking original node position which is necessary to evaluate matrix entries. The rearrange process for founding the bijective map is explained in the following with a 2D example. Color lines in Figure 2 refer to the judge line cross the centroid of cluster locate in different level of hierarchical structure mentioned in Section 3.1, and red points are the discretized nodes. The numbers around the red point denote the node indexes in original and current, the latter are contained in brackets.

The process for obtaining the block cluster tree recurs from the root of index cluster tree with pair subsets in x, y dimensions, and will not stop until the Equation (12) condition is satisfied or comes across leaf subset in either dimensions. The final hierarchical block clusters structure can be referred to Figure 3. Through the partition process, all sub-blocks can be identified two parts: admissible and non-admissible. A block is called admissible as it refers to a cluster of integration elements, whose distance, from the cluster of source nodes is about a certain threshold (see Equation (12)); or

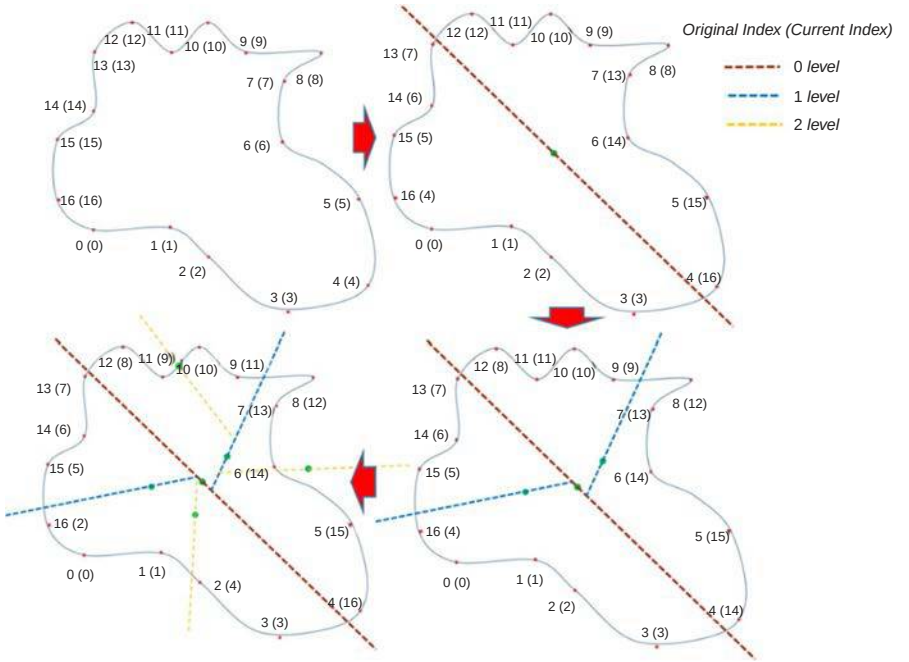

Figure 2.
Rearrange process for node index in 2D

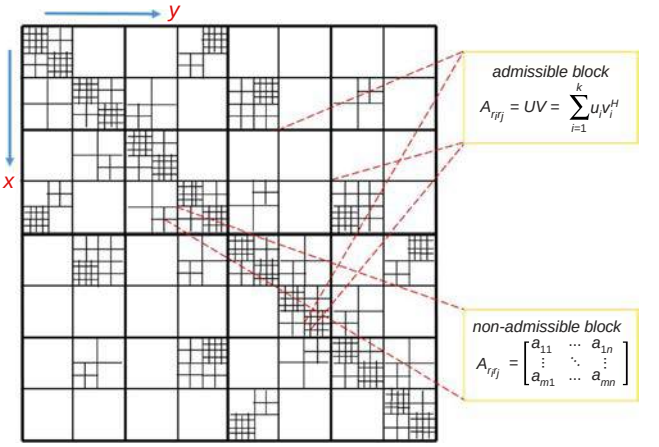

Figure 3.
Hierarchical matrix structure with block clusters

non-admissible as either cluster in pair of subset of cluster tree corresponding the block is smaller than *blocksize* n_{min}:

$$min\left(diam \cdot \Omega_{x_0},\ diam \cdot \Omega_x\right) \leqslant \eta \cdot dist\left(\Omega_{x_0},\ \Omega_x\right) \tag{12}$$

where Ω_{x_0} denotes the cluster contained source points corresponding to the row indices of a block in the original matrix and Ω_x the set of integration elements corresponding to the column index.

All admissible block could admit a compressed low-rank approximation. The compressed low-rank approximation of the blocks theoretically stems from the decomposition of kernel functions and the mathematical derivation is presented in Bebendorf's (2008a) work.

3.2 $\mathscr{H}$-Matrices arithmetic

This section takes a brief introduction about basic arithmetic operations with $\mathscr{H}$-Matrices such as addition, matrix-vector production, truncation and MMP. The details of expression could refer to Hackbusch's (1999) work. The inversion operation will be discussed detail in next section. In the following, the symbol $\mathscr{H}(T, k)$ refers to the $\mathscr{H}$-Matrix with maximum rank k, T represents the cluster partitions for the indices of all nodes, $\mathscr{L}(T_{J\times K})$ refers all leaves in the partition set, $F_j(t)\in T_I$ and $F_j(s)\in T_K$ denote the uniquely defined ancestors of t and s from the jth level of partition T_I and T_K, respectively.

Truncation. To avoid the increase of storage and computation complexity with improved rank value of block, it needs to set a preliminary fixed small value as maximum block rank. Truncation is an approximation procedure for mapping $\mathrm{H}(T, k')$ to $\mathrm{H}(T, k)$ with decreasing k' to fixed maximum rank k without modifying hierarchical structure. For each sub-block in hierarchical matrix, only low-rank block represented in outer-product formulation (Bebendorf, 2008a) should be truncated during specific arithmetic operations between $\mathscr{H}$-Matrices. ACA method is applied for approximating admissible blocks by out-product format in this paper.

Matrix-vector multiplication (MVP). MVP is the most popular arithmetic operation in numerical problems, and researchers have proposed several kinds of fast algorithms

for acceleration in past century, such as FMM, fast fourier transform method and so on. $\mathcal{H}$-Matrices arithmetic is another technique for accelerating this process from $O(N^2)$ to almost linear complexity. Because of the low-rank blocks in hierarchical matrix, the data referring to production process will decrease greatly, which leads the improvement of efficiency for the multiplication by matrix and vector. The number of operations N_{MV} required for MVP A*b = R of A∈H($T_{I\times J}$, k) by a vector b holds that:

$$N_{MV} \sim \max\{k, n_{\min}\}[|I|\log|I| + |J|\log|J|] \tag{13}$$

To be brief, this operation procedure actually accumulates the product results, obtained by multiplying each leaf in $\mathcal{H}$-Matrix with corresponding vector values (as Figure 4 shows), into each corresponding position of objective matrix level by level.

Addition. This operation in $\mathcal{H}$-Matrices always combines truncation operation mentioned above, which means this process is much more complicate than the usual entrywise matrices addition procedure. The formatted addition in $\mathcal{H}$-Matrix is defined as a truncation of the sum to the set of $\mathcal{H}$-Matrices (Grasedyck and Hackbusch, 2003). Then the operation complexity performed in each low-rank block is about $O(k_A+k_B)^2(m+n)$ when the dimension of block is $m\times n$ and ranks are k_A and k_B, respectively. For avoiding this time-consuming operation, MHDM is proposed in next section.

MMP. The MMP is $O(N^3)$ arithmetic operations and one of its main issues about time consumption during numerical computation. Actually, MMP is taken as group of MVP operations in traditional method, and each matrix data would be recalled repeatedly. So those effective algorithms for MVP, such as FMM, are no longer good at the MMP operation. In Wei *et al.* (2012) work, FMM was introduced for elastodynamic simulation, and parallel algorithm was involved to compensate low efficiency.

With applying $\mathcal{H}$-Matrices, this procedure will be transformed into the multiplication between leaf blocks and then executes the superposition level by level. Due to the same reason with MVP, less scale of data makes the computation efficiency improved significantly. The detail processes for MMP is described in Equation (14), the result is obtained from accumulating each block production results from root level to the final level:

$$(AB)_{ts} = \sum_{j=0}^{l} \sum_{r \in U_j(t\times s)} A_{tr}B_{rs} \tag{14}$$

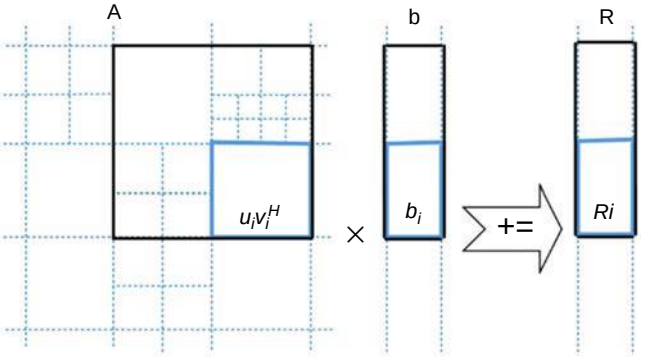

Figure 4.
Matrix-vector multiplication

where $U_j(t \times s) := \{r \in T_j^{(j)}: F_j(t) \times r \in T_{I \times J}$ and $r \times F_j(s) \in L(T_{J \times K})$ or $F_J(t) \times r \in L(T_{I \times J})$ and $r \times F_J(s) \in T_{J \times K}\}$

For example, there is three level in hierarchical matrices in Figure 5, dotted line framework denotes partition structure. The leaf block in root level 0 is first searched and child level 1 will be processed when the leaf set in level 0 is empty. There are three leaves in level 1 in both A and B matrices, such as ① block, which means we could find the set $U_j(t \times s)$ as $j = 1$. As figure shows, A_{tr} and B_{rs} corresponds the yellow parts of block ①, and the production result will add to the final corresponding position. We should note that the hierarchical structure of multiplication result is determined by both two input hierarchical matrices structure.

4. Application of $\mathscr{H}$-Matrix and DRBEM

4.1 Inversion algorithms in serial and parallel

Matrix inversion arithmetic is a huge time-consumption procedure in numerical operations, and also absolutely necessary for free-vibration elastodynamics using DRBEM. The complexity of inversion is about $O(N^3)$ with traditional methods with N dimension matrix. During the regularization evaluation process, each row in matrix is not independent with index during computation. There is still little effective algorithm proposed for acceleration of the procedure except for some approximation techniques. With applying hierarchical characteristics in $\mathscr{H}$-Matrices, serial recursive inversion algorithm is presented in Table I, the final inverse result is archived from agglomeration sub-results level by level. For example, the exact inverse value of the block cluster M on level p is obtained from block clusters on level p-1, as Equation (15) shows:

$$M^{-1} = \begin{bmatrix} {M_{11}}^{-1}\left(I + M_{12}S^{-1}M_{21}{M_{11}}^{-1}\right) & -{M_{11}}^{-1}M_{12}S^{-1} \\ -S^{-1}M_{21}{M_{11}}^{-1} & S^{-1} \end{bmatrix} \tag{15}$$

where:

$$S = M_{22} - M_{21}{M_{11}}^{-1}M_{12}$$

As for the pseudo-code in Table I, capital letters denote matrix data in any block, S means the son sets partition of current block, m, n, l and r denote dimension annotate in hierarchical block, and the subscript refers to the partition index. For example, $\mathrm{AUX}_{r_i \times r_j}$ denotes the block data of r_i dimension length in row and r_j dimension length in column. In the following description, AUX is used as auxiliary matrix to store temporary data, M is the input matrix, and INM is the inversion result.

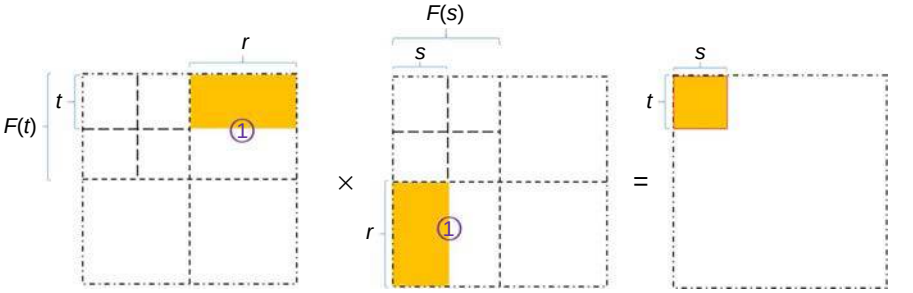

Figure 5.
Illustration for MMP procedure

```
Procedure  Inverse_HMatrix(var M, int r, var InM, var Aux)
           if (S(r×r) = ∅) then InM_{r×r} := (M_{r×r})^{-1}
           else determine the sons of r set, S(r) = {r1, r2}
              for i = 1:2 do
                 Inverse_HMatrix(M, r_i, InM, Aux);
                 if ((j = i − 1) > 0) then Aux_{r_i×r_j} := 0;
                    MultiAdd_HMatrix (Aux_{r_i×r_j}, r_i, r_i, r_j, InM_{r_i×r_i}, InM_{r_i×r_j});
                    InM_{r_i×r_j} := Aux_{r_i×r_j};
                 if ((j = i+1) < 3) then Aux_{r_i×r_j} := 0;
                    MultiAdd_HMatrix(Aux_{r_i×r_j}, r_i, r_i, r_j, InM_{r_i×r_i}, InM_{r_i×r_j});   M_{r_i×r_i} := Aux_{r_i×r_j};
                 if ((k = i+1) < 3) then
                    for j = 1:2 do
                       Aux_{r_k×r_j} := 0; MultiAdd_HMatrix(Aux_{r_k×r_j}, r_k, r_i, r_j, M_{r_k×r_i}, InM_{r_i×r_j});
                       Aux_{r_k×r_j} := − Aux_{r_k×r_j}; Add_HMatrix(InM_{r_k×r_j}, r_k, r_j, InM_{r_k×r_j}, Aux_{r_k×r_j});
              end
              if ((j = i+1) < 3) then
                 Aux_{r_k×r_j} := 0; MultiAdd_HMatrix(Aux_{r_k×r_j}, r_k, r_i, r_j, M_{r_k×r_i}, InM_{r_i×r_j});
                 Aux_{r_k×r_j} := − Aux_{r_k×r_j}; Add_HMatrix (M_{r_k×r_j}, r_k, r_j, InM_{r_k×r_j}, Aux_{r_k×r_j});
           end
           for i = 1:2 do
              if ((j = i − 1) > 0) then
                 for k = 1:2 do
                    Aux_{r_j×r_k} := 0; MultiAdd_HMatrix(Aux_{r_j×r_k}, r_j, r_i, r_k, M_{r_j×r_k}, InM_{r_i×r_j});
                    Aux_{r_j×r_k} := − Aux_{r_j×r_k};
                    Add_HMatrix (InM_{r_j×r_k}, r_j r_k, Aux_{r_j×r_k}, Aux_{r_j×r_k});
           end end

Function   MultiAdd_HMatrix(var C, int m, int l, int n, var A, var B)
           C′ := 0 ∈ R^{m×n}
           if (S(m×l) = ∅ or S(l×n) = ∅) then
              C' = A_{m×l}×B_{l×n};
              Add_HMatrix(C,m,n,C, C');
           else for each m'∈S(m), l'∈S(l), n'∈S(n) do
                 MultiAdd_HMatrix(C'_{m'×n'}, m', l', n',A,B);

Function   Add_HMatrix(var C,int m,int n,var A,var B)
           if (S(m×n) = ∅) then
              C_{m×n} = C_{m×n}+A_{m×n}+B_{m×n};                    //truncation addition
           else for each m'×n'∈S(m×n) do Add_HMatrix(C, m', n',A,B);
```

Table I.
Pseudo-code for matrix inverse algorithm in $\mathscr{H}$-Matrix structure

Comparing with traditional method, $\mathscr{H}$-Matrices method for inversion computation indeed has great effect on improving computation efficiency. Nevertheless, for the approximation procedure while obtaining hierarchical matrices, the relative error with approximation data must be transformed and enlarged along with recursive inversion operations. Besides the serial algorithm for inversion operation discussed in detail, this paper also proposed a parallel algorithm with CUDA for accelerating the operation. Figure 7 takes a new parallel inversion algorithm with CUDA and would not involve additional approximation error. The comparison of efficiency with the two inversion algorithm is presented at the end of this section.

With the dependent relationship during rows in matrix, the inversion procedure could not simply separate all the serial internal operations into different kernels. Figure 6

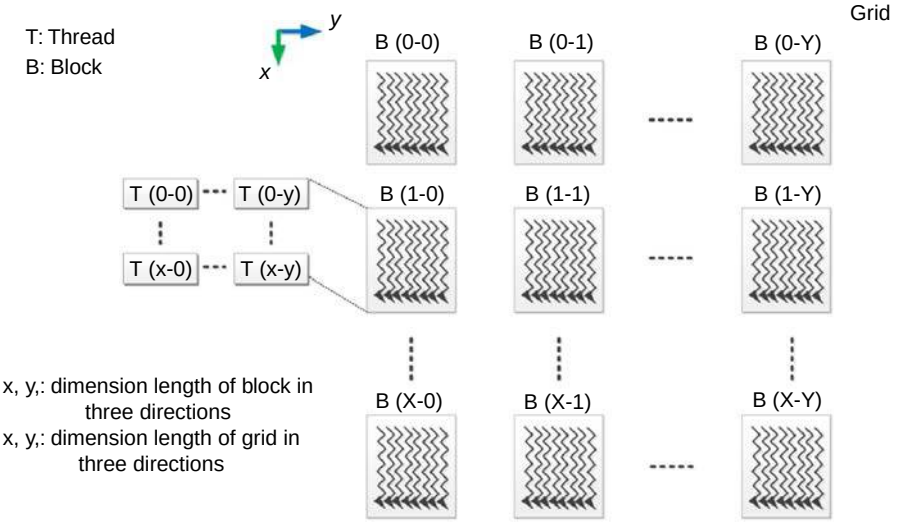

Figure 6.
Parallel structure of CUDA in GPU

introduces the distribution of kernels for parallel inversion algorithm in GPU. The top module is grid contains blocks (as the symbol B refers in Figure 6) in X and Y direction, and the number is determined by the total number of threads needed for computation and capacity of each block. Each block contains numbers of threads (as the symbol T refers in Figure 6), and the final kernel function will be executed in these threads. The threads in block also have the direction separation with x and y.

During the parallel algorithm introduction in Figure 7, T means the thread position in global threads, which also corresponds to data position in matrix data in this algorithm the suffix _x (_y) implies the position in x (y) direction; the suffix _D refers to the diagonal data in matrix, and _C is the data in fixed column.

For testing the effect of algorithms mentioned above, we select five different number of degrees of freedom (DOFs), such as 1,356, 3,348, 4,995, 7,257, 10,584, to count the time and memory consuming. We should know that the scale of matrix is directly depends the number of DOFs. The speedup results with respect to different DOFs are summarized in Figure 8 with comparison between serial hierarchical matrices algorithm and parallel algorithm, and the memory consumption results are presented in Figure 9. The suffix _M in figures denotes the host memory consuming and _DM means the device memory consuming.

It is easy to find that the time would rise rapidly in serial algorithm as the scale of problem increasing, and the increment in parallel algorithm is much gentler, as Figure 8(a) shows. In current computational environment, the speedup ratio is about 13 times in average for parallel algorithm and just about five times for $\mathscr{H}$-Matrices according to Figure 8(b). Therefore, the conclusion can be obtained that the parallel inversion algorithm with CUDA has obvious advantage in efficiency with respect to $\mathscr{H}$-Matrices strategy. However, from Figure 9, the memory-consuming comparison indicates parallel method would consuming much more storage space in host memory than either traditional serial method or $\mathscr{H}$-Matrices strategy, and it also needs device memory which is unnecessary for other two methods. The display memory space could not be added manually as usual host memory, and the space is generally limited by 1-2 GB in personal computer at present. So the scale of solvable problem would be limited if the parallel

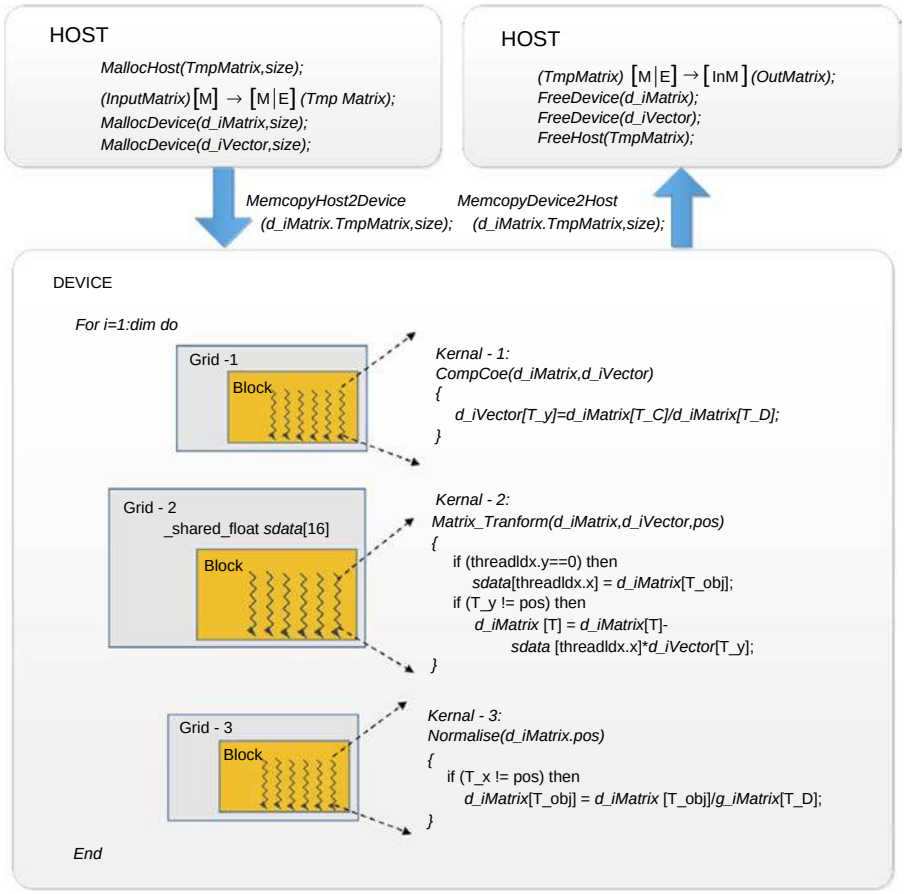

Figure 7.
Inverse algorithm in parallel with CUDA

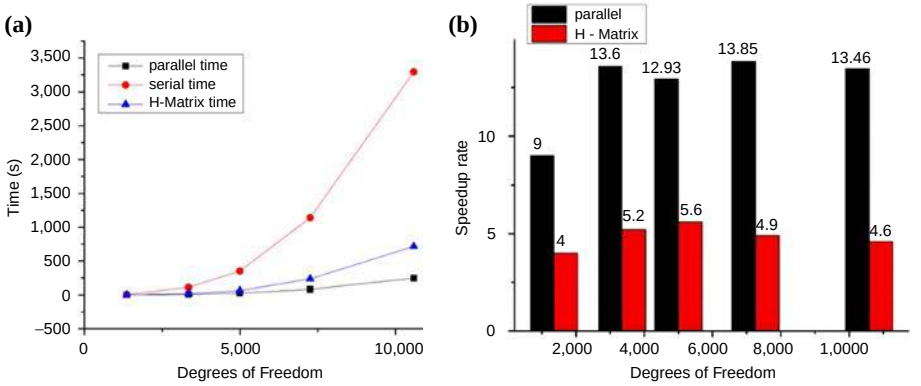

Figure 8.
Time-consuming and speedup rate for different inversion method

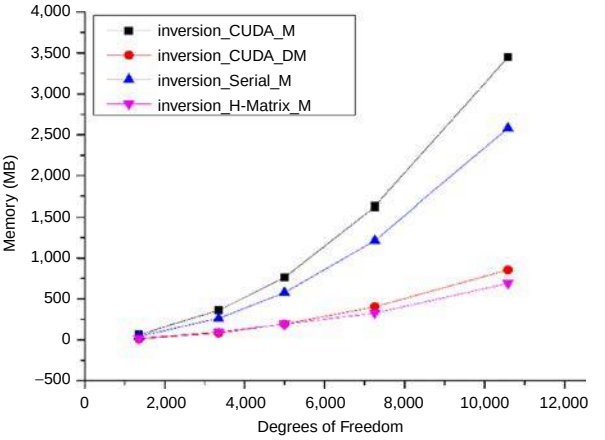

Figure 9.
Memory consuming in memory and display card memory for different method

inversion algorithm is applied for $\mathscr{H}$-Matrices&DRBEM. We also could find that $\mathscr{H}$-Matrices strategy has obvious advantage with respect to traditional methods in memory consumption. Thus, it is reasonable to apply $\mathscr{H}$-Matrices inversion algorithm for large-scale problems and parallel algorithm for high-efficiency demand.

4.2 PHDM and MHDM

At the beginning of this section, we will first discuss the treatment for the singular Cauchy principal value (CPV) integration during Equation (3) while obtaining the coefficient matrix $\boldsymbol{H}$. The usual method for dealing with this singular integration is rigid-body motion method which cannot be applied in hierarchical matrices for incomplete matrix entries. The approximation method proposed by Guiggiani and Gigante (1990) for CPV integrals on surface domain is introduced for evaluating the entries located in diagonal of hierarchical structure. The singular integration would be separated into two parts: local area e_ε in which the source point ξ is at a distance $\leqslant\varepsilon$, and non-singular integral parts of field elements. And the integration should be converted into polar coordinates in local parameter plane of field element. The detail processes of this method can refer to the works by M. Guiggiani and A. Gigante.

With respect to the equation of DRBEM presented in the second section, the main time cost issues are MMP and inversion operation. As the introduction in the above, we should know it could save memory space and improve computation efficiency by introduction $\mathscr{H}$-Matrices for MMP operation. On the other hand, MMP contains truncation addition arithmetic which is the high-time consuming part while evaluating Equation (4). Although some strategies for truncation, such as orthogonality method (Bebendorf, 2008a), have been investigated, the improvement for the efficiency is slightly, especially for large-scale problems.

During the investigation of MMP operation in $\mathscr{H}$-Matrices, we find that the multiplication between $\mathscr{H}$-Matrices and entrywise matrix will not lead to additional truncation operation, which means it is possible to accelerate production procedure by using two different types of matrix for avoiding truncation operation. Two different strategies are proposed in following based on matrix structure. One is covering all the matrices referred in numerical equation, such as coefficient matrices and particular solution matrices, with hierarchical structure for maximum compressing storage

consumption named PHDM. This strategy could make the memory space consumption decreasing to 10-30 percent over the traditional method. However, this kind of method could not guarantee the highest efficiency for truncation operation. So another strategy for purchasing efficiency performance is presented which at the cost of some storage space. During the MMP operation, only one of multiplier matrices is denoted with hierarchical structure and the other will be represented with traditional entrywise structure. This kind of consideration is called MHDM. The detail processes for these two methods applied for DRBEM is described in Figure 10.

Because the $\boldsymbol{H}$ matrix need to participate follow up MMP operations, as Equation (4) implies, which is usually represented with entrywise structure in MHDM algorithm. Then the singular integration appeared in $\boldsymbol{H}$ matrix could be calculated by rigid-body motion method. The $\boldsymbol{M}$ matrix for inversion operation while solving natural frequencies also need to maintain entrywise structure, so it could use high-effective parallel inversion algorithm proposed in the above.

According to the description in the Figure 10, it is obvious that MHDM will consume much more memory space for full populated entrywise matrices $\boldsymbol{P}'$, $\boldsymbol{W}'$, $\boldsymbol{GP}'$, $\boldsymbol{HW}'$ and $\boldsymbol{S}'$, and also will archive much more superior efficiency performance at the same time. The comparison for efficiency and memory consuming will discussed in the last section with examples.

5. Numerical examples

This section validates the accuracy and efficiency of $\mathscr{H}$-Matrices strategy proposed above through several numerical examples. The linear triangular elements are used in both examples. All the program codes are executed on a desktop computer with Intel Core 2 I7 CPU, and the host memory is 8 GB. The display card is NVIDIA GeForce 590 which owns 1.5 GB device memory space. The peak floating-point arithmetic performance is 4.64 Tflops and 984 Gflops in single and double precision, respectively. In examples, the minimal *blocksize* $n_{min} = 15$, $\eta = 0.5$ and the accuracy for ACA approximation $\varepsilon = 10^{-4}$ are chosen, and the material parameters are: Shear modulus $\mu = 10^6$ Pa, Poisson's ratio $v = 0.3$, and mass density $\rho = 7{,}400\ \mathrm{kg/m^3}$.

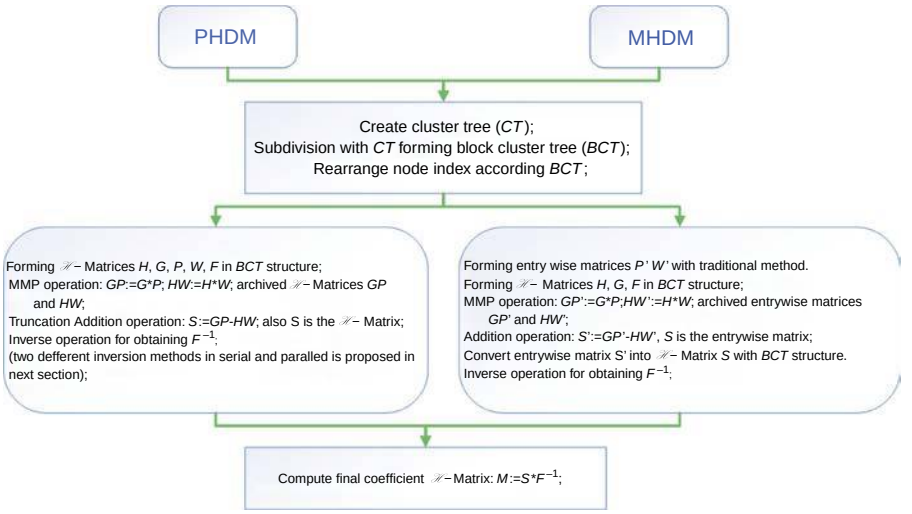

Figure 10.
Different strategies for DRBEM with $\mathscr{H}$-Matrix

5.1 Example 1

Some symbols will be applied for simplicity in tables and figures, which represent the following quantities:

- DOFs: degrees of freedom.
- T: time consuming.
- HM: host memory consuming.
- DM: device memory consuming.
- SR: speedup rate.
- R: ratio of the amount of memory consumed.
- M/Tra: ratio of the amount of time consumed by traditional and MHDM.
- P/Tra: ratio of the amount of time consumed by traditional and PHDM.

In this example, the shaft model is involved for testing time and memory consumption with the proposed methods in this paper. The consumption in time and memory space from traditional DRBEM method and FMM algorithm are also presented for comparison.

The shaft is discretized into six different number DOF as Figure 11 shows. The time-consumption and memory space requirement for host and device will not stop counting until the coefficient matrix $\boldsymbol{M}$ in Equation (6) is achieved, and the peak value will be recorded as result for comparison in this section.

The Table II presents the detail situation for time-consuming and speedup ratio in different DOF with respect to PHDM, MHDM, FMM and traditional DRBEM, Figure 12 reveals the discipline and tendency at the same time. We can conclude that the former two methods have obvious advantage with respect to traditional DRBEM in efficiency either in time or in memory aspects. As the DOF added, the time needed for traditional DRBEM and FMM increases rapidly while the other two methods grow gently, which could be observed from A-1 in Figure 12. We can also observe that MHDM performs better in efficiency than PHDM, and the effect will be much more obviously as the DOF increasing. At the same time, the time comparison of CPU and CUDA with HDM computation is listed in A-1, and the advantage of CUDA could be clearly observed. For

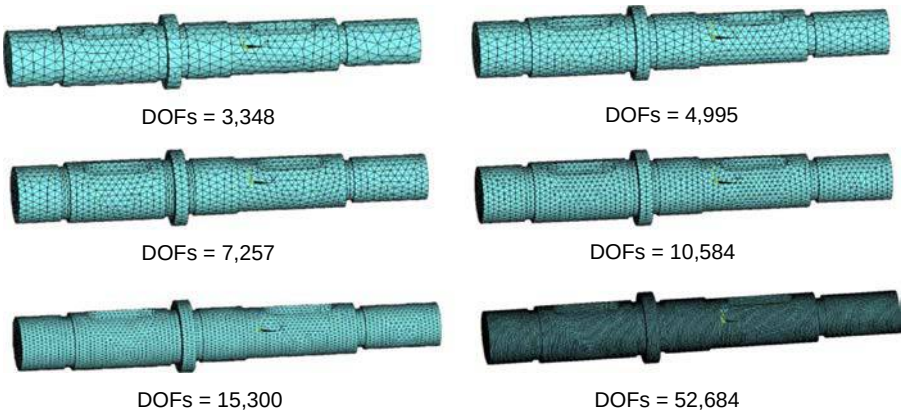

Figure 11.
Shaft with different number degrees of freedom

DOFs	3,348	4,995	7,527	10,584	15,300	52,684
Traditional						
T(min)	16.632	58.668	183.85	576.47	/	/
HM(MB)	438.7	1,012.43	2,106.5	4,519.05	/	
FMM						
T(min)	15.637	78.92	216.15	928.3	2741	/
HM(MB)	48.32	61.246	96.47	142.39	205.4	911.83
PHDM						
T(min)	4.941	11.241	25.91	78.197	159.57	1,699.4
SR	3.366	5.219	7.09	7.372	/	/
HM(MB)	156.2	362.8	556.8	832.1	1,044.3	4,137.9
R	0.356	0.358	0.264	0.18	/	/
MHDM						
T(min)	2.432	6.719	17.27	48.246	/	/
SR	6.839	8.732	10.346	11.949	/	/
HM(MB)	298.7	658.46	1,353.3	2,843.5	/	/
R	0.68	0.65	0.64	0.62	/	/
DM(MB)	87.5	201.3	411.2	874.2	/	/
MHDM-CPU						
T(min)	3.757	8.413	20.975	55.46		
SR	4.427	6.973	8.765	10.394		
PHDM-CPU						
T(min)	6.431	13.694	29.115	84.583	227.28	2,473.4
SR	2.586	4.284	6.315	6.815		

Table II.
Detail data for time and memory consuming

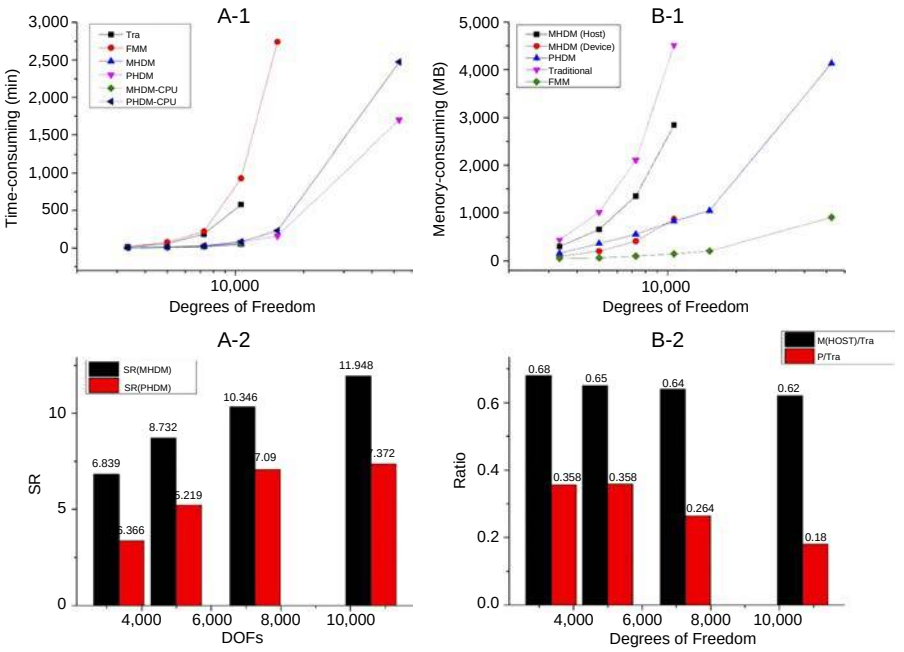

Figure 12.
Comparison of time-consuming and speedup rate in different degrees of freedom

explaining the effect of acceleration for CUDA method, the speedup ratio are presented in A-2 of Figure 12 (the detail analysis of the inversion efficiency solved by CPU and CUDA, respectively, can refers to Section 4). MHDM apparently has superior speedup ratio with respect to PHDM. With the increasing of the DOF, the speedup ratio of MHDM with respect to different DOFs grow almost linearly from 6.839, 8.732, 10.346 to 11.949. However, PHDM's speedup ratio grow much gentler from 3.366, 5.219, 7.09, to 7.372. It is worth noting that because of MMP computation during simulation process, FMM will consume much more time even comparing with traditional method (the time consuming is too long to be listed in table while the DOF up to 52,684).

Besides the efficiency, memory-consuming situation is also noticed in this example. It can be easily observed that the methods proposed in this paper could save lots of space while evaluating final results. The peak consuming of host and device memory in each method is listed in B-1 of Figure 12, and the ratios of host memory consumption of novel methods by traditional DRBEM are also presented in B-2. The storage requirement of PHDM is just about one-third of traditional DRBEM, and would decrease to about one-fourth with the increment of DOF. Under the same situation, MHDM's requirement for host memory is about half of traditional DRBEM, and the ratio would also decrease with the increment of DOF. However, MHDM will consume device memory space for applying parallel inverse algorithm at the same time, which limit the scale of solvable problems. For this reason, the limitation of DOF involved with MHDM in our computer is about 14,000, and the capability for PHDM is about 1.0E5. The memory consuming of FMM algorithm is also presented. According the result data, it can be found that FMM would occupy very small amount of storage space with respect to other methods, which is a big advantage of this algorithm. But comparing the methods proposed in this paper, its efficiency is too low to be involved for dealing with large-scale problems.

5.2 Example 2

Due to the approximation in each admissible cluster while introducing hierarchical structure for improving efficiency and saving memory consuming, the relative error will be involved and transferred to the final results. In this paper, the relative error $\in$ is obtained according to Equation (16), HDM denotes the result data from proposed method (PHDM or MHDM) and Tra refers to traditional DRBEM.

This example is discussing the accuracy of natural frequencies obtained through methods mentioned in this paper, and the results from ANSYS and Hyperworks in the same situation are presented for comparison. In the following, we also simply discuss the effect of the number of internal point (IP) for the accuracy. The pedestal body, as the Figure 13 shows, will be discretized into different DOFs for this example:

$$\varepsilon = \frac{HDM - Tra}{Tra} \tag{16}$$

The fore ten order natural frequencies with MHDM, PHDM, traditional DRBEM is listed in Figure 14, and same for Ansys and Hyperworks. According to Figure 14(a) and (b) are the result comparison with different methods in 5,076 and 10,336 DOFs, respectively. It can be found that the gap for proposed methods, with respect to the results obtained from FEMs, is bigger than the traditional DRBEM. With increasing the DOFs, the gap would be narrowed, which can be observed from the contrast in

Figure 14(a) and (b). The frequencies of fourth, sixth, seventh order in PHDM, and of fourth, fifth, ninth order in MHDM, are obviously getting closer to the results in Ansys and Hyperworks with the number of DOFs increasing.

Figure 14(c) and (d) involve different number of IPs, 101 and 214, respectively, for discussing the effect for accuracy by IP in DRBEM. The figure **c** has obviously nice performance except for the eighth order frequencies which has a big shift with respect to Anasys's and Hyperworks's results. As the IPs improved, Figure 14(d) gets more superior performance than all the other situations. Although the number of IPs indeed has positive effect for the accuracy in general, but this kind of effect is not immovable. Actually, the accuracy of results evaluated with DRBEM is not bounded to the number of IPs, the accuracy will greatly increases for a small increment

Figure 13.
Pedestal body for computing natural frequencies and model

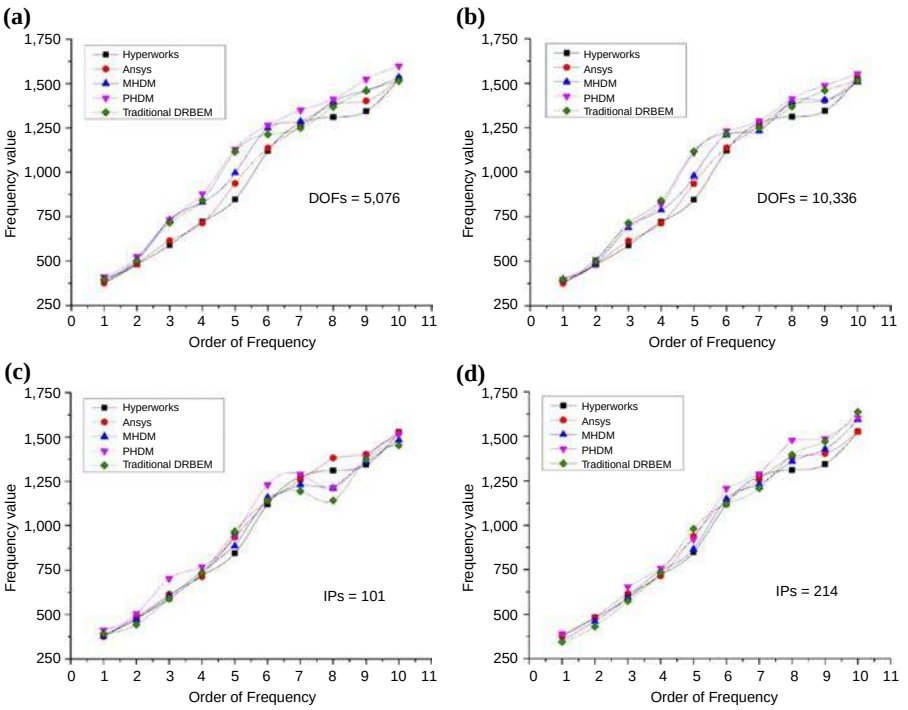

Figure 14.
Comparison of natural frequencies in different situations and methods

of number of IPs but decreases as introducing too much. The detail description of choosing collocation points refers to Chirino *et al.* (1994) and Agnantiaris *et al.* (1996), in which the view that a small number of collocation points improve the accuracy of the solution is proved.

The relative error in natural frequencies corresponds to traditional DRBEM for proposed strategies are presented in Figure 15. M_RE refers to MHDM relative error and P_RE refers to PHDM. It is obvious that P_RE performs much more stable than M_RE, and becomes more gently as the DOF increasing. The computation between hierarchical matrix and entrywise matrix would cause instability error, as the specific high value in fifth shows. It worth noticing that for high-quality and uniform mesh procedure, the accuracy would be slightly affected by the number of DOFs and also the relative error would not decreased dramatically through increasing the number of DOFs.

6. Conclusion

For solving free-vibration problems, DRBEM has higher efficiency than traditional time-domain or frequency-domain BEM. In this paper, the approximation partition technology $\mathscr{H}$-Matrices with ACA is introduced for accelerate free-vibration analysis procedure. PHDM and MHDM strategies are presented and discussed in terms of both time and space complexity regarding to various computation situations. The former strategy applied $\mathscr{H}$-Matrices structure into all involved integration matrices, which ensures the maximum compression of memory requirement and improves the efficiency with respect to traditional method. Then, some other situations demand high efficiency and accuracy are considered by MHDM which utilizes CUDA techniques to guarantee inversion operation efficiency, then performs arithmetic between different types of matrices for avoiding complicate truncation operation. Also the serial recursive $\mathscr{H}$-Matrices inversion algorithm are introduced in PHDM to accelerate the computation process (the effect comparison of the two inversion method is discussed in Section 4). The time and memory consumption of serial examples are counted in the last section, and compared with traditional and FMM methods to verified the effect of method. The accuracy of the strategies proposed in this paper is also discussed by detail comparison with commercial software. In the future, the parameters' effect, such as n_{min}, η during the partition process in hierarchical matrices, will be discussed with respect to efficiency and accuracy. The possibility about hierarchical matrices arithmetic in parallel formulation will be also studied.

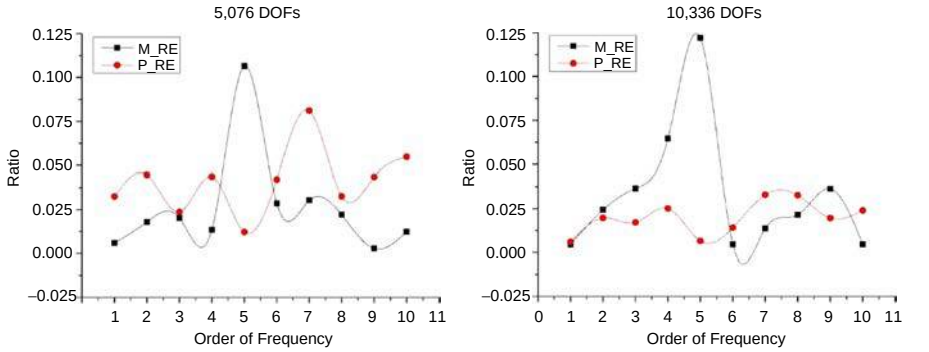

Figure 15.
Relative error with respect to traditional DRBEM

References

Agnantiaris, J.P., Polyzos, D. and Beskos, D.E. (1996), "Some studies on dual reciprocity BEM for elastodynamic analysis", *Computational Mechanics*, Vol. 17 No. 4, pp. 270-277.

Agnantiaris, J.P., Polyzos, D. and Beskos, D.E. (2001), "Free vibration analysis of non-axisymmetric and axisymmetric structures by the dual reciprocity BEM", *Engineering Analysis with Boundary Elements*, Vol. 25 No. 9, pp. 713-723.

Ahmad, S. and Banerjee, P. (1986), "Free vibration analysis by BEM using particular integrals", *Journal of Engineering Mechanics*, Vol. 112 No. 7, pp. 682-695.

Aparinov, A.A. and Setukha, A.V. (2010), "Application of mosaic-skeleton approximations in the simulation of three-dimensional vortex flows by vortex segments", *Computational Mathematics and Mathematical Physics*, Vol. 50 No. 5, pp. 890-899.

Banaugh, R. and Goldsmith, W. (1963), "Diffraction of steady acoustic waves by surfaces of arbitrary shape", *The Journal of the Acoustical Society of America*, Vol. 35, p. 1590.

Banerjee, P., Ahmad, S. and Manolis, G. (1986), "Transient elastodynamic analysis of three-dimensional problems by boundary element method", *Earthquake Engineering & Structural Dynamics*, Vol. 14 No. 6, pp. 933-949.

Bazilevs, Y., Calo, V.M., Cottrell, J.A., Evans, J.A., Hughes, T.J.R., Lipton, S., Scott, M.A. and Sederberg, T.W. (2010), "Isogeometric analysis using T-splines", *Computer Methods in Applied Mechanics and Engineering*, Vol. 199 No. 5, pp. 229-263.

Bebendorf, M. (2000), "Approximation of boundary element matrices", *Numerische Mathematik*, Vol. 86 No. 4, pp. 565-589.

Bebendorf, M. (2005), "Hierarchical LU decomposition-based preconditioners for BEM", *Computing*, Vol. 74 No. 3, pp. 225-247.

Bebendorf, M. (2008a), *Hierarchical Matrices*, Springer, Berlin Heidelberg.

Bebendorf, M. (2008b), "Hierarchical matrices: a means to efficiently solve elliptic boundary value problems", *Computational Science and Engineering*, Vol. 63, pp. 36-58.

Bebendorf, M. and Grzhibovskis, R. (2006), "Accelerating galerkin BEM for linear elasticity using adaptive cross approximation", *Mathematical Methods in the Applied Sciences*, Vol. 29 No. 14, pp. 1721-1747.

Bebendorf, M. and Rjasanow, S. (2003), "Adaptive low-rank approximation of collocation matrices", *Computing*, Vol. 70 No. 1, pp. 1-24.

Benedetti, I. and Aliabadi, M.H. (2010), "A fast hierarchical dual boundary element method for three-dimensional elastodynamic crack problems", *International Journal for Numerical Methods in Engineering*, Vol. 84 No. 9, pp. 1038-1067.

Chaillat, S.P., Bonnet, M. and Semblat, J.-F.O. (2009), "A new fast multi-domain BEM to model seismic wave propagation and amplification in 3-D geological structures", *Geophysical Journal International*, Vol. 177 No. 2, pp. 509-531.

Chirino, F., Gallego, R., Saez, A.S. and Dom Nguez, J. (1994), "A comparative study of three boundary element approaches to transient dynamic crack problems", *Engineering Analysis with Boundary Elements*, Vol. 13 No. 1, pp. 11-19.

De Mey, G. (1976), "Calculation of eigenvalues of the Helmholtz equation by an integral equation", *International Journal for Numerical Methods in Engineering*, Vol. 10 No. 1, pp. 59-66.

Dominguez, J. and Roesset, J.M. (1978a), "Dynamic stiffness of rectangular foundations", *NASA STI/Recon Technical Report N*, Vol. 79, p. 16152.

Dominguez, J. and Roesset, J.M. (1978b), "Response of embedded foundations to travelling waves", *NASA STI/Recon Technical Report N*, Vol. 79, p. 16141.

Grasedyck, L. (2005), "Adaptive recompression of matrices for BEM", *Computing*, Vol. 74 No. 3, pp. 205-223.

Grasedyck, L. and Hackbusch, W. (2003), "Construction and arithmetics of H-matrices", *Computing*, Vol. 70 No. 4, pp. 295-334.

Greengard, L. and Rokhlin, V. (1997), "A fast algorithm for particle simulations", *Journal of Computational Physics*, Vol. 135 No. 2, pp. 280-292 (reprinted from the (1987) *Journal of Computational Physics*, Vol. 73, pp. 325-348.).

Guiggiani, M. and Gigante, A. (1990), "A general algorithm for multidimensional Cauchy principal value integrals in the boundary element method", *Journal of Applied Mechanics*, Vol. 57 No. 4, pp. 906-915.

Gumerov, N.A. and Duraiswami, R. (2008), "Fast multipole methods on graphics processors", *Journal of Computational Physics*, Vol. 227 No. 18, pp. 8290-8313.

Hackbusch, W. (1999), "A sparse matrix arithmetic based on h-matrices. Part I: introduction to h-matrices", *Computing*, Vol. 62 No. 2, pp. 89-108.

Hackbusch, W. and Nowak, Z.P. (1989), "On the fast matrix multiplication in the boundary element method by panel clustering", *Numerische Mathematik*, Vol. 54 No. 4, pp. 463-491.

Hut, J.B. and Piet (1986), "A hierarchical O(N log N) force-calculation algorithm", *Nature*, Vol. 324 No. 6096, p. 4.

Karabalis, D.L. and Beskos, D.E. (1984), "Dynamic response of 3-D rigid surface foundations by time domain boundary element method", *Earthquake Engineering & Structural Dynamics*, Vol. 12 No. 1, pp. 73-93.

Karabalis, D.L. and Beskos, D.E. (1985), "Dynamic response of 3-D flexible foundations by time domain BEM and FEM", *International Journal of Soil Dynamics and Earthquake Engineering*, Vol. 4 No. 2, pp. 91-101.

Kurz, S., Rain, O. and Rjasanow, S. (2002), "The adaptive cross-approximation technique for the 3D boundary-element method", *IEEE Transactions on Magnetics*, Vol. 38 No. 2, pp. 421-424.

Liu, Y.J., Mukherjee, S., Nishimura, N., Schanz, M., Ye, W., Sutradhar, A., Pan, E., Dumont, N.A., Frangi, A. and Saez, A. (2011), "Recent advances and emerging applications of the boundary element method", *Applied Mechanics Reviews*, Vol. 64 No. 3, pp. 1-38.

Michel, B.D.E. (1987), *Boundary Element Methods in Mechanics*, WILEY-VCH Verlag, North-Holland.

Milazzo, A., Benedetti, I. and Aliabadi, M.H. (2012), "Hierarchical fast BEM for anisotropic time-harmonic 3-D elastodynamics", *Computers & Structures*, Vol. 96, pp. 9-24.

Nardini, D. and Brebbia, C.A. (1983), "A new approach to free vibration analysis using boundary elements", *Applied Mathematical Modelling*, Vol. 7 No. 3, pp. 157-162.

Nishimura, N. (2002), "Fast multipole accelerated boundary integral equation methods", *Applied Mechanics Reviews*, Vol. 55, p. 299.

Ostrowski, J., Andjelic, Z., Bebendorf, M., Cranganu-Cretu, B. and Smajic, J. (2006), "Fast BEM-solution of laplace problems with H-matrices and ACA", *IEEE Transactions on Magnetics*, Vol. 42 No. 4, pp. 627-630.

Park, H.-S. and Dang, X.-P. (2010), "Structural optimization based on CAD-CAE integration and metamodeling techniques", *Computer-Aided Design*, Vol. 42 No. 10, pp. 889-902.

Qin, X., Zhang, J., Li, G., Sheng, X., Song, Q. and Mu, D. (2010), "An element implementation of the boundary face method for 3D potential problems", *Engineering Analysis with Boundary Elements*, Vol. 34 No. 11, pp. 934-943.

Rizos, D.C. and Karabalis, D.L. (1998), "A time domain BEM for 3-D elastodynamic analysis using the B-spline fundamental solutions", *Computational Mechanics*, Vol. 22 No. 1, pp. 108-115.

Rizzo, F.J. (1967), "An integral equation approach to boundary value problems of classical elastostatics", *Quarterly of Applied Mathematic*, Vol. 40, pp. 83-95.

Rokhlin, V. (1985), "Rapid solution of integral equations of classical potential theory", *Journal of Computational Physics*, Vol. 60 No. 2, pp. 187-207.

Tai, G.R. and Shaw, R.P. (1974), "Helmholtz-equation eigenvalues and eigenmodes for arbitrary domains", *The Journal of the Acoustical Society of America*, Vol. 56, p. 796.

Tyrtyshnikov, E. (1996), "Mosaic-skeleton approximations", *CALCOLO*, Vol. 33 Nos 1-2, pp. 47-57.

Wang, L. (2009), "Integration of CAD and boundary element analysis through subdivision methods", *Computers & Industrial Engineering*, Vol. 57 No. 3, pp. 691-698.

Wang, Y., Wang, Q., Wang, G., Huang, Y. and Wang, S. (2013), "An adaptive dual-information FMBEM for 3D elasticity and its GPU implementation", *Engineering Analysis with Boundary Elements*, Vol. 37 No. 2, pp. 236-249.

Wei, Y.X., Wang, Q.F., Wang, Y.J. and Huang, Y.B. *(*2012), "Optimizations for elastodynamic simulation analysis with FMM-DRBEM and CUDA", *Cmes-Computer Modeling in Engineering & Sciences*, Vol. 86 No. 3, pp. 241-273.

Zhang, J.M. and Tanaka, M. (2007), "Adaptive spatial decomposition in fast multipole method", *Journal of Computational Physics*, Vol. 226 No. 1, pp. 17-28.

Zhang, J., Qin, X., Han, X. and Li, G. (2009), "A boundary face method for potential problems in three dimensions", *International Journal for Numerical Methods in Engineering*, Vol. 80 No. 3, pp. 320-337.

Zhang, J., Zheng, X., Lu, C., Xie, G. and Li, G. (2013), "A geometric mapping cross approximation method", *Engineering Analysis with Boundary Elements*, Vol. 37 No. 12, pp. 1668-1673.

Appendix

$$\kappa_{ij}^{m}=\frac{1}{4\mu(1-\nu)}\left[(3-4\nu)\left(\frac{r^2}{6}+\frac{r^3}{12}\right)+\frac{r^2}{10}+\frac{r^3}{18}\right]\delta_{ij}-\frac{1}{4\mu(1-\nu)}\left(\frac{2}{15}+\frac{r}{12}\right)r_i r_j, \tag{A1}$$

$$\begin{aligned}\varsigma_{ij}^{m}=&\frac{1}{2(1-\nu)}\left[(1-2\nu)\left(\frac{1}{3}+\frac{r}{4}\right)+\frac{1}{5}+\frac{r}{6}\right]\left(r_i n_j+r_k n_k\delta_{ij}\right)\\&-\frac{1}{2(1-\nu)}\left[(1-2\nu)\left(\frac{1}{3}+\frac{r}{4}\right)-\frac{1}{5}-\frac{r}{6}\right]r_j n_i-\frac{1}{2(1-\nu)}\cdot\frac{1}{12r}r_i r_j r_k n_k,\end{aligned} \tag{A2}$$

r_i is the components of the vector r that connecting any two points of boundary nodes or interior points, and n_i is the components of the normal outward vector n at the point that the particular solution is evaluated, and δ_{ij} is the Dirac function.

Corresponding author
Professor Yunbao Huang can be contacted at: huangyb@mail.hust.edu.cn