Copyright © 2012 Tech Science Press CMES, vol.86, no.3, pp.241-273, 2012

Optimizations for Elastodynamic Simulation Analysis with FMM-DRBEM and CUDA

Yixiong Wei[1], Qifu Wang[1,2], Yingjun Wang[1] and Yunbao Huang[1]

Abstract: In this study, we propose a novel method to accelerate the process of elastodynamic analysis in 3D problems with BEM (boundary element method). With applying the DRBEM (dual reciprocity boundary element method) to form new integral equations for reducing complexity;the modified FMM (fast multipole method)is introduced to simplify the computation process and save storage space by avoiding intermediate coefficientmatrices. At the same time, FMM-DRBEM is reprogrammed in parallel byapplying GPU with CUDA (Compute Unified Device Architecture)for improving efficiency further.The main features in this paper are: (1)with respect to defects of classical method for elastodynamic, modified FMM-DRBEM algorithm is introduced; (2)optimal algorithms of translationsin FMM are discussed in parallel for improving the effect of GPU with CUDA;(4) PLE (Position Location Equation) and GHST (Global Hierarchy Substitution Transferring) methods are proposedfor improving efficiency, saving storage space requirement and optimizing data transforming process between host and device, respectively. The effect has been tested by numerical examples in the last section, and significantoptimal results both in efficiency and accuracy have been observed.

Keywords: FMM-DRBEM, CUDA, parallelize algorithm, 3D elastodynamic problems

1 Introduction

Because ofthe characteristics such as dimensiondecomposition, high precision, theBEM is more suit for fastpreprocessing, self-adaptive structure analysis engineering applications than other numerical methods. Nevertheless, forthe low stability and high cost in time, classical BEM is rarely used for solving 3D elastodynamic problems in engineering[Liu; Mukherjee; Nishimura; Schanz; Ye; Sutradhar; Pan; Dumont;

[1] National CAD Support Software Engineering Research Center, Huazhong University of Science and Technology, Wuhan, Hubei, China 430074
[2] Corresponding Author. E-mail Address: wangqf@hust.edu.cn

Frangi and Saez (2011)]. The non-symmetric and fully populated matrix of BEM results in huge workload for computation in elastodynamics. Furthermore, the integral equation will need furtherdiscretization for time domain for elastodynamic problems. Suppose that an elastodynamics boundary value problem for a cube uses$O(n)$ nodes and $O(m)$ time steps.N is the scale of coefficient matrix. In this case, the computational complexity of BEM is $O(n^4)*O(m^4)$ since $N=3*n^2*3*m^2$, but only $O(n^3)*O(m)$ for FDM (Finite difference method) and FEM (Finite element method)at the same time.

FMM was introduced by Rokhlin [Rokhlin (1985)] as an O(N)numerical method for solving an integral equation with 2D Laplace's equation. This method becomes well noted with introducing into N-body problems by Greengard [Greengard and Rokhlin (1987; Greengard (1988)] for solving potential and particle simulation problems.Many authors applied FMM for elastodynamic problems to improve efficiency and save storage space. Chen [Chen; Chew and Zeroug (1997)]extends the high-frequency FMM in Helmholtz proposed by Rokhlin [Rokhlin (1990)] to 2D elastic wave surface formulation. Fukui [Fukui (1998)] introducelow-frequency FMM algorithm into 2D scattering problems for many holes, and which was also applied to solve scattering problems for multi cavities and cracks by Fujiwara [Fujiwara (1998)] in corresponding. In 3D field, Yoshida [Yoshida (2001)] utilized low-frequency FMM to solve low-frequency crack problems with decomposition of fundamental function, and also proposed diagonal form FMM for elastodynamic crack problems in 2001[Yoshida; Nishimura and Kobayashi (2001)]. The recent development in elastodynamic FMM is slow until the diagonal form redesigned by Chaillat et.al.[Chaillat; Bonnet and Semblat (2007)].Besides FMM algorithm, other acceleration methodology was applied such as pre-corrected FFT (Fast Fourier Translation) approach.[Yan; Zhang and Ye (2010)], and the H-matrix (ACA: Adaptive Cross Algorithm)approach in crack problems[Benedetti and Aliabadi (2010)].

Ahead of FMM, DRBEM(Dual Reciprocity Boundary Element Method) had already beenintroduced by Nardini and Brebbia[Nardini and Brebbia (1983)] since 1983.Chirino et.al [Chirino; Gallego; Sáez and Dominguez (1994)] have concluded that the DRBEM requires less computing time than either the time or frequency domain of conventional BEM's. During DRBEM algorithm, the inertial volume integral in BEM equationis transformed into a surface integral by applying the reciprocal theorem twice, andthe displacement fieldis approximatedby a finite series variables involving RBF (radial basis functions)[Agnantiaris; Polyzos and Beskos (2001)]. This method combines the advantages of dimensionality reductionand simplicity of elastostatic foundation solution. Ahmad and Banerjee [Ahmad and Banerjee (1986)] proposed a particular integral BEM approach for conversion from domain to surface ones, which is proved to be mathematically equivalent to DRBEM

[Nardini and Brebbia (1983)].

The FMM-DRBEM applied in this research combines the characteristics of FMM and DRBEM. However, two disadvantages need to be solved before application. First, in contrast with the banded, sparse and symmetric matrix formed by FEM, plenty of non-symmetric and full populated matrix operations in DRBEM consumes huge memory space and CPU time; second, as the price of simplicity with elastostatic fundamental functions, some additional MMP (Matrix-Matrix production) calculation with $O(N^2)$ complexity are required, which have low efficiency duringcomputation.Thus, the CUDA parallel algorithm is designed in section 4.

Since the general application of CUDA in engineering industries,parallelize FMM algorithms are developed by many authors. Gumerovand Duraiswami[Gumerov and Duraiswami (2008)] pioneered fine-grained parallel FMM (Fast Multipole Method) algorithm based on CUDA. In this algorithm, RCR (rotation coaxial-translation rotation) decomposition and translation stencil techniques were applied for optimization. Compared to traditional coarse-grained parallel algorithm based on CPU, the three dimensional Laplace kernel FMM achieved a speed up of 30 to 70 times on a single NVIDIA GPU[Gumerov and Duraiswami (2008)].

Inheriting the basic algorithm from Gumerov and Duraiswami, Yokota applied FMM with cluster of GPUs for meshless simulation of turbulence. The speed up ratio on a single GPU is about 80 times than using two CPUs. And the parallel efficiency improves with the increase of particle number, e.g. the parallel efficiency in 32 GPU is just 4% at $N = 10^4$, but 78% at$N = 10^7$; this work was investigated by Hamadaet al. in conjunction with their high-performance GPU implementation of a tree code[Hamada; Narumi; Yokota; Yasuoka; Nitadori and Taiji (2009)].

Cruz et.al discussed the optimization strategies for GPU kernels with FMM as example[Cruz; Layton and Barba (2011)]. In the related work, the author introduced four properties to judge the algorithm in GPU model — computational intensity, concurrency, homogeneity, and data locality, which were taken as criteria in his paper. On the other hand, this paper assessed each stage in the forming processes of GPU kernel, such as the partition of threads, share memory use, memory management and so on. Those design strategies are very helpful and efficient while modifying CPU algorithm into GPU kernels.

The speed up ratio of each translation procedure is discussed and the effect of function expansion order is also exhibited in detail. During the processes of FMM, the M2L (Moment to Local Expansion translation) and near interaction computation take almost 80% time. Takahashi studied M2L operation and presented four algorithms to accelerate the process[Takahashi; Cecka; Fong and Darve (2012)]. Although each algorithm has different characteristic and functions, the key point of

his work is the improvement of reusability of coefficient data in M2L translation. This algorithm transforms the 64 MVP (Matrix-Vector production) computations into 27-MMP computations, in which most data could reuse in computation. However, the total complexity actually increases from $512r'*r^2$ to $1728r'*r^2$ (r' and r is the order of series expansion). In connection with this situation, we propose a new data structure to decrease the complexity for M2L optimization.

In this paper, the secondand third sections briefly introduce the DRBEM and conventional FMM theories;then the modified algorithm in FMM is explained in detail. The fourth section introduces the optimal parallel translation algorithms in detail; the effectof our research is demonstrated in the last section with numerical examples.

2 DRBEM inElastodynamics

It can simplify the process of computation greatly 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 as

$$(\lambda+2\mu)\nabla\nabla\cdot\mathbf{u}(\mathbf{x},t)-\mu\nabla\times\nabla\times\mathbf{u}(\mathbf{x},t)+\rho b=\rho\ddot{\mathbf{u}}(\mathbf{x},t), \tag{1}$$

in which $\mathbf{u}(\mathbf{x},t)$ is the displacement vector at point $\boldsymbol{x}$ and time t, and λ and μ are Lame elastic constants and ρ is density of body. Assuming zero body forces and initial conditions, as applying traditional BEM, one can obtain an integral representation of the Eq. (1) in the form

$$\mathbf{c}(x)u(x,t)=\int_{\Gamma}[\mathbf{U}^*(x,t;;\xi,0)*p(\xi,t)-\mathbf{W}^*(x,t;;\xi,0)*u(\xi,t)]d\Gamma(\xi), \tag{2}$$

in which $\mathbf{U}^*(x,t;;\xi,0)$ and $\mathbf{W}^*(x,t;;\xi,0)$ is the dynamic fundamental solutions of displacement and traction tensor in respect, $\mathbf{p}(\xi,t)$ is the traction vector at point ξ and time t. ***c(x)*** is the jump tensor. The fundamental solution is very complicated for computation. Because of the time variables in integral equation, the complexity will be $O(n^4)*O(m^4)$compare to the $O(n^3)*O(m)$in FEM with $O(n)$ nodes and $O(m)$ time steps, that is very inefficiency. For simplifying the computation process, one

can substitute static functional solutions with dynamic ones, such as

$$\mathbf{c}(x)\mathbf{u}(x,t) = \int_{\Gamma} [\mathbf{u}^*(x,\xi)\mathbf{p}(\xi,t) - \mathbf{p}^*(x,\xi)\mathbf{u}(\xi,t)]d\Gamma(\xi) - \int_{\Omega} \mathbf{u}^*(x,\xi)\rho\ddot{\mathbf{u}}(\xi,t)d\Omega(\xi), \quad (3)$$

in which $\mathbf{u}^*(x,\xi)$ and $\mathbf{p}^*(x,\xi)$ are static fundamental solutions and the overdots indicate differentiation with respect to time.However, the appearance of the inertial volume integral in Eq. (4) indicates the discretization in volume domain is necessary, which would eliminate the biggest advantage of BEM. Nardini and Brebbia [Nardini and Brebbia (1983)] transferred this volume integral to the boundary surface, thereby creating an all-boundary integral formulation that leading to DRBEM.

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

$$u_i(x,t) = \int_{m=1}^{M} \alpha_i^m(t) f^m(x), \; x \in \Omega, \quad (4)$$

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. Using the reciprocity theorem again, one succeeds in transforming Eq. (4) into a boundary integral form

$$-\int_{\Omega} \mathbf{u}^*(\mathbf{x},\xi)\rho\ddot{\mathbf{u}}(\xi,t)d\Omega(\xi) = \rho\int_{m=1}^{M} \ddot{\alpha}_n^m \left[c_{ij}(\mathbf{x})\kappa_{jn}^m(\mathbf{x}) + \int_{\Gamma} p_{ij}^*(\mathbf{x},\xi)\kappa_{jn}^m(\mathbf{x})d\Gamma(\xi) - \int_{\Gamma} u_{ij}^*(\mathbf{x},\xi)\zeta_{jn}^m(\mathbf{x})d\Gamma(\xi)\right], \quad (5)$$

in which $\kappa_{jn}^m(\mathbf{x})$ and $\zeta_{jn}^m(\mathbf{x})$ are the particular solutions for displacement and traction field, respectively. Then,with the discretization of the boundary Γ into numbers of triangle elements (the total number of node is N), Eq. (4) could form the matrix equation

$$[\mathbf{M}]\{\ddot{\mathbf{u}}\} + [\mathbf{H}]\{\mathbf{u}\} = [\mathbf{G}]\{\mathbf{p}\}, \quad (6)$$

where

$$[\mathbf{M}] = \rho([\mathbf{G}][\mathbf{P}] - [\mathbf{H}][\mathbf{W}])[\mathbf{F}]^{-1}, \quad (7)$$

in which [$\boldsymbol{H}$] and [$\boldsymbol{G}$] are the integral coefficient $N \times 3 \times N \times 3$matrices, and [$\boldsymbol{P}$] and [$\boldsymbol{W}$] are matrices containing sub-matrices of particular solution κ_j^m and ζ_j^m each

column of which corresponds to the m-order radial function and each row refers to the jth nodal point. When introducing the collocation points to improve accuracy, the matrices could be wrote as

$$[\mathbf{M}] = \rho \left(\begin{bmatrix} \mathbf{G}_{\mathbf{B}B} \\ \mathbf{G}_{\mathbf{D}B} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{\mathbf{B}B} & \mathbf{P}_{\mathbf{B}D} \end{bmatrix} - \begin{bmatrix} \mathbf{H}_{\mathbf{B}B} & \mathbf{H}_{\mathbf{B}D} \\ \mathbf{H}_{\mathbf{D}B} & \mathbf{H}_{\mathbf{D}D} \end{bmatrix} \begin{bmatrix} \mathbf{W}_{\mathbf{B}B} & \mathbf{W}_{\mathbf{B}D} \\ \mathbf{W}_{\mathbf{D}B} & \mathbf{0} \end{bmatrix} \right) [\mathbf{F}]^{-1}, \tag{8}$$

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 Eq. (6). The frequency domain equation is

$$\left(-\omega^2 [\mathbf{M}] + [\mathbf{H}]\right) \{\mathbf{u_0}\} = [\mathbf{G}] \{\mathbf{p_0}\}, \tag{9}$$

in which ω is the circular frequency of the harmonic excitation of $\boldsymbol{u}$ and $\boldsymbol{p}$vectors with amplitude $\mathbf{u_0}$ and $\mathbf{p_0}$, respectively. Just setting the external disturbances equal to zero, one can obtain natural modes and frequencies of free vibration.

$$[\mathbf{A}] \{\mathbf{x}\} = \omega^2 [\mathbf{M}^*] \{\mathbf{x}\}, \tag{10}$$

in which [$\boldsymbol{A}$] is the BEM influence matrix referring to all unknown boundary variables contained in $\{\mathbf{x}\}$, and $[\mathbf{M}^*]$ is obtained by setting zeros in [$\boldsymbol{M}$] in which sub-columns refer to specified displacements. From the work of J.P. Agnantiaris [Agnantiaris; Polyzos and Beskos (2001)] for 3D elastodynamics, the effect of augmentation in the linear radial basis function for accuracy is very small, and for non-axisymmetric 3D structures polynomial 1+r is simplest and high accuracy RBF. Therefore, the corresponding particular solution equation for displacement and traction can be achieved in respect.[Agnantiaris; Polyzos and Beskos (1998)]

$$\kappa_{ij}^m = \frac{1}{4\mu(1-\nu)} \left[(3-4\nu) \left(\frac{r^2}{6} + \frac{r^3}{12} \right) + \frac{r^2}{10} + \frac{r^3}{18} \right] \delta_{ij} - \frac{1}{4\mu(1-\nu)} \left(\frac{2}{15} + \frac{r}{12} \right) r_i r_j, \tag{11}$$

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

in which μ is the shear modulus, ν the Poisson's ratio, 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.

3 Fast Multipole Method in DRBEM

The fast multipole method can be considered as a method which provides a way to compute the multiplication between matrix and vector without intermediate matrix. Hence, this method is very suitable for solving potential and particle simulation problems. The conventional MVP algorithm needs $O(N^2)$ complexity in general, which is too expensive to large-scale problems. By the contrast, the FMM provides the matrix-trial vector product $O(N)$ operations, which is a huge improvement in efficiency.

Comparing with general data structure, we take STL (standard template library) to simplify construction and explain the structure of all relationship in FMM; the fundamental C language is applied during all translation computations to improve efficiency. In this section, the translations of kernel functionare brief reviewed at first, more detail information can be found in Nishimura's work [Nishimura (2002)]. Then the new version of FMM algorithm is featured up in detail, which applies new data structure (*node-elementcell* structure) and algorithms for improving searching efficiency in the procedure of computing the expansion of kernel function.

3.1 Basic Functions Introducing

According to the character of DRBEM, we just need considering the elastostatic fundamental solution for elastodynamics problem. The fundamental functions, expressed as$U_{ij}(x,y)$ and $T_{ij}(x,y)$, can be expanded by solid harmonic functions which are expressed by the associated Legendre function [Yoshida (2001)].In the Appendix, Eq.(a1) and Eq. (a2)denote these expansions. Therefore, the integration equations can be rewrote by these expansions, and the direct integration between points substituted by the intermediate integrations and kinds of translation.

During these translations, the transferring from source to expansion point is called moment expansion, and the corresponding transferring from filed to expansion point is called local expansion which depend the results of moment expansion. Yoshida[Yoshida (2001)] provides details of these expansions which can be referred in Appendix.

These formulas are derived from results in Epton MA [Epton MA (1995)] in terms of the Wigner-3j symbols and concluded by Yoshida [Yoshida (2001)].

3.2 Description of Modified FMM

1. *level*: the division step;
2. *cell*: the box that contains node or element;
3. *limit-number*: the minimal limit number of node in each cell.

As the concepts proposed by Rokhlin [Rokhlin (1985)], the surface elements will be divided into different *cells* in different *level*s after the meshing procedure for the surface of body, as Figure 1 shows. In this study, the index number of *cell* is labeled by the division order. For example, the *cell* labeled i in *level* l are divided into m child *cells* which are labeled from t to t+m (t is the number of cells in *level* l+1 at moment). And while these child *cells* are *leaves* (the number of points in the *cell* is zero or less than *limit-number*), the *cell* i+1 of *level* l could be divided and the child *cells* will be labeled from t+m+1, and so on, as shows in Figure 1.

There will be two kinds of *cell* data structure in oct-tree: one is named *NCell* which stores point data (include nodes and collocation points), and another is named*ECell* which stores element data. In our algorithm, the number of *NCell* in current *level* is stored in vector *CellNumL*, similarly, the *ECell* is stored in vector *CellNumM*. The total node number in mesh data is expressed as *NodeNum*, collocation point number is expressed as *Con*, and element number is expressed as *ElemNum*.

In our algorithm, the current *levelcells* are instantiated in each recursion procedure, and each *cell* object contains the pointer of parent *cell* and child *cells*. The recursion will not end until all the*cells* are *leaf*, then the total number of *level* will be counted as *levelnum*. As mentioned in the above sub-section, the local expansion depends on the moment expansion of each *cell*, and the moment value of each *non-leafcell* depends on moment value of *leafcell*. Therefore the process of FMM is separated as *upward* (for moment expansion) and *downward* (for local expansion) procedures.

In upward, the moment computation in *leafcell* is named E2M (element to moment) and the moment of *non-leaf cell* is evaluated by M2M (moment to moment). The downward procedure is carrying on from *level*2 to *levelnum*-1, and contains translations of M2L (moment to local expansion) between two *cells* in same *level*, L2L (local to local expansion) between parent and child *cells*, NIC (near interaction computation) between contiguous*cells*, and FLE (final local expansion) during the leaf *cells*.

Because of the difference in L2L and NIC translation, the distance of target *cell* and source *cell* should be evaluated for determination in traversal *cell* process. However, this strategy is very inefficient, because the number of contiguous *cells* are at most 27 (contain itself), and the number of interaction *cells* (the parent *cells* is con-

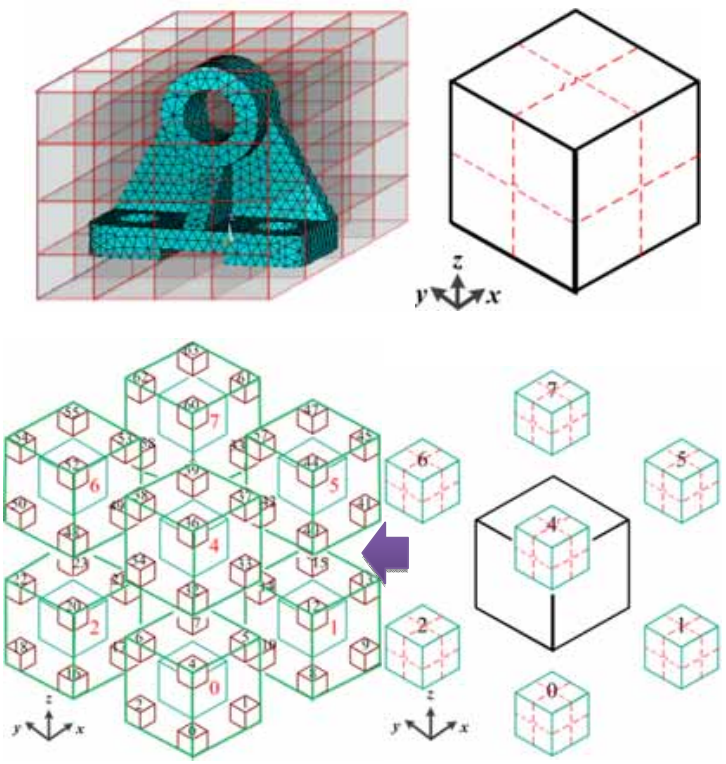


Figure 1: The forming procedure of *cells* in *level*s

tiguous but not for child *cells*) are at most 189, See Figure 2 gray and purple *cells*, respectively.We derive the Position Location Equation (PLE) (See Eq. (12)) to locate the neighbor and interaction *cells* position with respect to target *cell* without traversal of all *cells*. The complexity is at most 27 for NIC and 189 for M2L.

$$Position = (z+m)*2^{Div}+(y+n)*Div+x+l, \tag{13}$$

$$\begin{cases} 0<(z+m)<(Div/2-1) \\ 0<(y+n)<(Div/2-1) \\ 0<(x+l)<(Div/2-1) \end{cases}$$

in which

$$z = i/2^{Div}$$

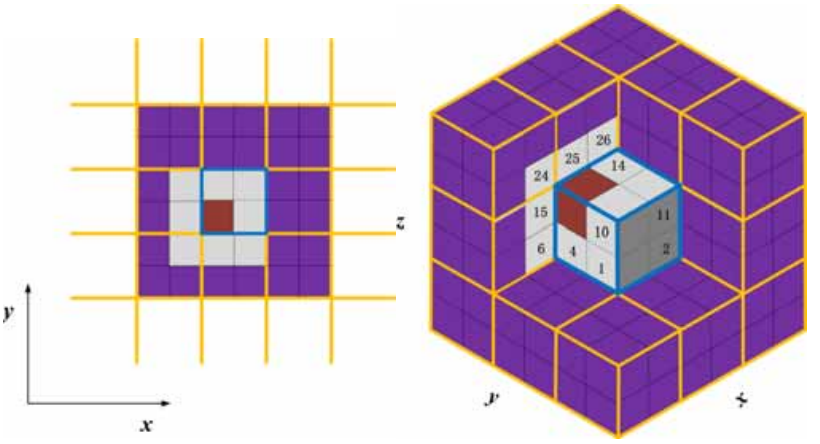


Figure 2: Indication of relationship between target and source *cells*

$$y = (i\%2^{Div})/Div$$

$$x = \left(i\%2^{Div}\right)\%Div$$

Div is the numbers of division in current *level*, and *m,n,l* is neighbor *cell* position (front or back, left or right, up or down) with respect to target *cell*, with value of 1,0 or1.

In this research, another modified aspect for FMM algorithm is proposed. In numerical solutions, the DRBEM integrations would be discretized with format as follows:

$$Pc_{ij}(\mathbf{x})\,u_j(\mathbf{x},\mathbf{s}) = \sum_{n=1}^{N_e}\sum_{\alpha} t_j^{n\alpha}(\mathbf{s})\int_{-1}^{1}\int_{-1}^{1} u_{ij}^{*}(\mathbf{x},\ \mathbf{y}(\xi_1,\xi_2),\mathbf{s})N_\alpha(\xi_1,\xi_2)J^n(\xi_1,\xi_2)\,d\xi_1 d\xi_2 - \sum_{n=1}^{N_e}\sum_{\alpha} u_j^{n\alpha}(\mathbf{s})\int_{-1}^{1}\int_{-1}^{1} t_{ij}^{*}(\mathbf{x},\ \mathbf{y}(\xi_1,\xi_2),\mathbf{s})N_\alpha(\xi_1,\xi_2)J^n(\xi_1,\xi_2)\,d\xi_1 d\xi_2, \quad (14)$$

in which N_e denotes number of elements contain the filed point $\boldsymbol{s}$; α denotes the filed point in different element; ξ_1, ξ_2 denote the parameters in local coordinate system of element.

In traditional way, the integration of Eq. (13) between source point and the element must be computed repeatedly while each node of the element is treated as field point. Actually, because the value in matrix is the contribution from point to point,

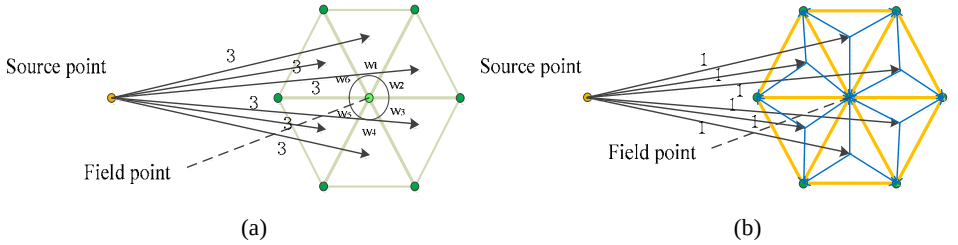


Figure 3: Different integration method

each point to element integration must repeats three times, as Figure 3 (a) shows for example. The integration result can be expressed as η, and w express the shape value of the correspond node.Then, the final value of correspond point to point integration is achieved by the following equation:

$$\eta_1 * w_1 + \eta_2 * w_2 + \cdots + \eta_6 * w_6.$$

Suppose the *ElemNum* is M, then the complexity of this kind of integration is $3M$. Hence, for reducing the redundant computation, we could evaluate all the integration between source points and elements at first, then transferring the value to all nodes in the matrix by shape functions, (See Figure 3 (b)).It could be obtained that this method's complexity is just M, which saves plenty of computation cost relative to traditional way.For realizing this method, we apply two sets of *cell* data to avoid repeated computation, one of sets stores *NCell* and the other stores *ECell*(mentioned in the above).

4 Parallelize in GPU

For convenient, some notations are listed in the Table 1.

Because the FMM for elastodynamic problems include MMP computation, the parallel algorithm must concurrent the rows of matrix B into threads of x dimension, which means each thread in x dimension evaluate the multiplication with one row of matrix B. The other two dimension states different intention in different kernel of algorithms, the details refer to Table 4.

In this section, Partition Reduction Summation method is introduced firstly, then the translations of FMM in parallelize are discussed. M2M and L2L have the same data structure; M2L applied a new algorithm to improve efficiency; E2M, FLE have the similar kernel structure. We should note that NIC translation needs far more space than other translations. If the total number of nodes exceeds 1500, the

Table 1: Some notations about the parallel algorithm

Notation	Description
NodeNumC	the number of nodes in *Ncell* box
ElemNumC	the number of elements in *Ecell* box
dim	*NodeNum* + *Con*
Nexp	the order of series expansion
TolNum	(*Nexp* + 1)*(*Nexp* + 2)/2
TolNum1	(*Nexp**2+1)*(*Nexp**2+2)/2
d_M	moment value in the *Ecell*
d_L	local translation value in the *Ncell*
Ejt	the translation operator between j *cell* and t *cell* during M2M, M2L and L2L translation

requirement space of NIC will be 2˜3 times than other translations, and this distance will increases with the scale of problem. For improving the scale of solvable problem, we do not parallel NIC translation with CUDA in GPU, but with OpenMP in CPU.

4.1 Partition Reduction Summation

During the introduction of FMM-DRBEM algorithm in the third section, it is worth noting that there are lots of summation computations while forming expansion values or final matrix. Just asFigure 3(b)shows, it needs to accumulate all coefficients of the node corresponds to each element which includes it. However, in the parallel process, this summation is easy leading to visit conflict in GPU memory (reading and writing the same place at same time). The general efficient summation algorithm in parallel is Reduction Summation (RS)[NVIDIA (2011)], which decreases the procedures of summation from N to $\sqrt{N}$ (N is the number of thread in one dimension of a block).

However, this method requests that all relative data are referred in one block that could be accessed in different thread. The number of nodes and elements in each *cell* is different and uncertain, and the difference may be very large in complicated body discretization. So it is impossible to set fixed length of block to evaluate the summation of interactions in E2M, FLE. This paper presented a new summation

Table 2: Estimate equation for translations in GPU memory (**MB**)

E2M	$\frac{96dim * TolNum * (1+\frac{E}{4})}{1024*1024}$	NIC	/
		M2M	$\frac{72dim * TolNum^2 + SM}{1024*1024}$
FLE	$\frac{12dim * TolNum * (8+N)}{1024*1024}$	M2L	$\frac{72dim * TolNum^2 + SM + SL}{1024*1024}$
		L2L	$\frac{72dim * TolNum^2 + SL}{1024*1024}$

N: *NodeNumC* in *leaf cell*; *E*: *ElemNumC* in *leaf cell; SM*: size of moments in *cells* of current *level*; *SL*: size of local expansion in *cells* of current *level*

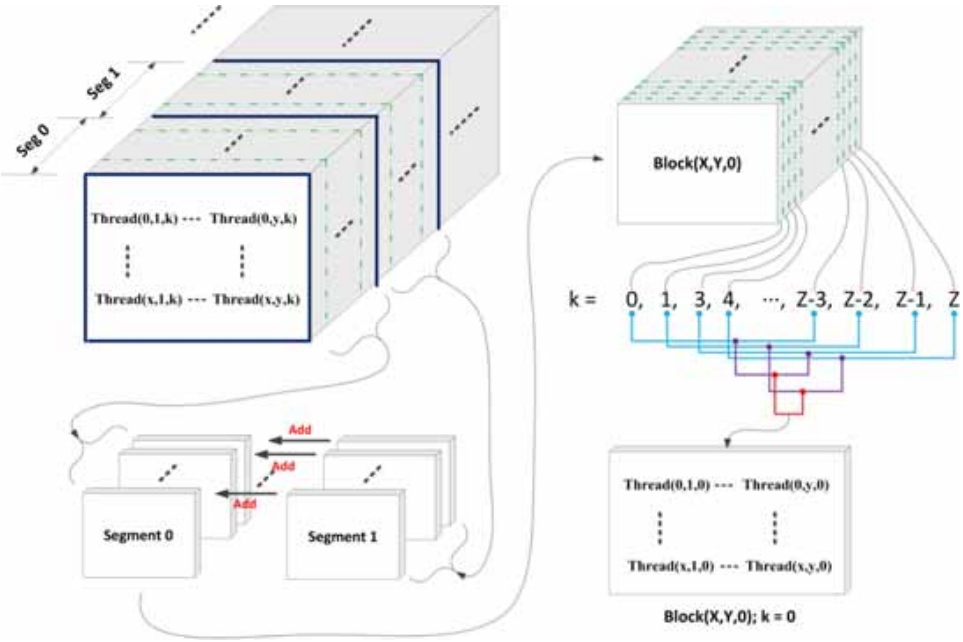


Figure 4: Partition Reduction Summation method

method called Partition Reduction Summation (PRS).

Firstly, the whole data is partitioned into N/M segments which have the same structure with a block; the data in each thread of segment will be added into segment 0, respectively. Then, the block will refer to segment 0 by thread ID and apply RS al-

gorithm to evaluate the final result, just as Figure 4 shows. Compared to traditional method, the procedure of summation could decrease to $N/\sqrt{M}$ (M is one dimension length of block), and this method will not be limited by dimension length of block.Through this method, the requirement storage space in GPU of each translation also reduces, and the estimate equation for each translation is listed in Table 2.

4.2 E2M, FLE and NIC

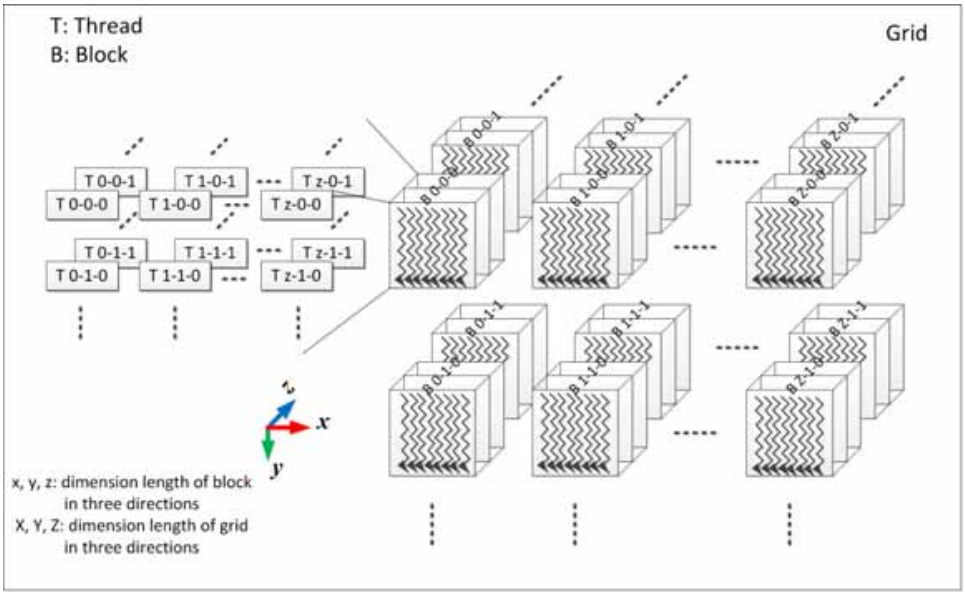


Figure 5: Arichitecture of kernel in translations

E2M In the traditional way, the moment expansion of each leaf *cell* is computed by traversing every *cell* in each *level*, which is not suit for parallel processes. A vector (*leaflist*) to store all*leafcells* pointer, and the moment of all*leaf* are evaluatedbefore translations of M2M and M2L is executed. Because of the limitation of storage space, the E2M translation is evaluated by circle, though the *cell* data is independent for each other. The pseudo code of E2M parallel algorithm could write as Table 3 shows.

The three dimensions of grid are expressed by *dim*, *ElemNumC* and *TolNum* as Table 4 shows. The architecture of kernel is described in Figure 5. For the parallel

algorithm, one thread executes one moment evaluation and data accessing with global memory in GPU. After the E2M translation of elements in GPU, the PRS is applied to achieve the final moment expansion of the *ECell*.

FLE and NIC The processes of parallel translation algorithm of FLE are similar to E2M, but the data structure has some differences. The FLE translation takes place in *leafcell* of tree structure, the results obtained by this procedure will be put into the final result matrix. By contrast with E2M, the Z-dimension direction is expressed with the order of series expansion (*TolNum*) which is the certain value defined in before. Because of that, the length of Grid in Z direction is 1, and RS algorithm is applied directly into the summation of threads in Z-dimension of a block. The pseudo codes of FLE are listed in Table 3.

In this study, the near interaction *cell* is determined by PLE presented in section 3.2. OpenMP in CPU is introduced to parallel this translation (NIC), and eight processor threads are applied. The pseudo code for the process is listed in Table 3.

Table 3: Pseudocode for translations

E2M translation	**FLE translation**	**NIC translation**
fori = *levelNum***To** 2	**for**i = *levelNum***To** 2	**for**i = *levelNum***To** 2
for j = 0 **To***CellNumM*[i]	**for** j = 0 **To***CellNumL*[i]	**for** j = 0 **To***CellNumM*[i]
E2Mkernel(···) --- GPU	Obtain near source *cell* k respect to *cell* j	**if***cell* j is leaf
PRS evaluation		#pragma omp parallel for
end for	FLEkernel(j, k, ···) --- GPU	**for** j = 0 **To***Nearcells*
end for	RS evaluation	NIC (···)---CPU
	end for	**end if**
		end for
	end for	PRS kernel
		end for
		end for

4.3 M2M, L2L

For the limitation of storage space, the moments and local expansion value of *level* can't transfer in one time. Considering this situation, the page-locked memory

Table 4: The index of three dimension of Grid in different translations

Direction / Translation	X-dimension	Y-dimension	Z-dimension
E2M	*dim*	*ElemNumC*	*TolNum*
M2M	*dim*	*TolNum*	*TolNum*
M2L	*dim*	*TolNum*	*TolNum*
L2L	*dim*	*TolNum*	*TolNum*
FLE	*dim*	*NodeNumC*	*ElemNumC*

of CUDA is applied in the algorithm. There are three kind of page-locked memory provided by CUDA, portable memory, write-combining memory and mapped memory[NVIDIA (2011)]. The first kind memory make any device (multi-GPU) could access this memory, which will reduce transferring and communication time in CPU threads. The second kind memory will frees up host's L1 and L2 cache resources, making more cache availableto the rest of the application. In addition, write-combining memory is not snoopedduring transfers across the PCI Express bus, which can improve transferperformance by up to 40%.[NVIDIA (2011)]Reading from write-combining memory from the host is prohibitively slow, sowrite-combining memory should in general be used for memory that the host onlywrites to. The mapped memory could be applied in out parallel algorithm. There is no need to allocate a block in device memory and copy data betweenthis block and the block in host memory; data transfers are implicitly performedas needed by the kernel. That means the scale of problems which could be paralleled in GPU will improve greatly. On the other hand, the efficiency of this method is lower than the GHST (Global Hierarchy Substitution Transferring) method proposed in this paper. Table 5lists the time cost of the two transfer method in different degree of freedom by discretization for a cubic body.

M2M The Moment to Moment translation (M2M) is used to evaluate the moment of parent *cell* respect to current *level*. In simple terms, that is accumulating all the child *cell* moment to parent *cell* with multiplying coefficient operator E_{tj}. Because the translation of M2M happens between current *level* and parent *level*, this paper employs GHST to save storage space and improve efficiency. As Figure 6 shows,

Table 5: Comparative time cost of translations in different data transfer method (***ms***)

DOFs -Transfer Method	Translations	M2M	L2L	M2L	FLE
1356	GHST	137	53	1937	28
	Mapped Memory	203	91	4019	44
3804	GHST	101	76	6160	119
	Mapped Memory	167	133	16128	246
10122	GHST	356	282	13672	1975
	Mapped Memory	594	511	29140	2216

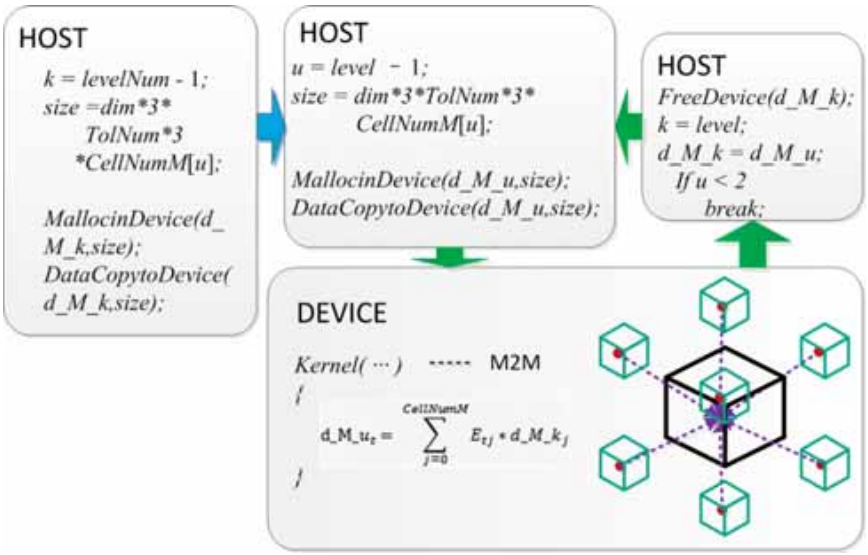


Figure 6: Global Hierarchy Substitution Transferring for M2M

except for the first step, each circulation in levels only needs transferring parent *level* moment data, the result evaluated in previous time would be reused in next time, and the*d_M_k*memory in device should be freed in before.

The method of global *level* moment data transferring in one time could avoids the repeated transfer during the computation in the kernel, which is the key point for achieving higher performance than mapped memory method. The hierarchy substitution transferring excludes the redundant moment data which would be useless in current *level* moment evaluation and saves the storage space of GPU.

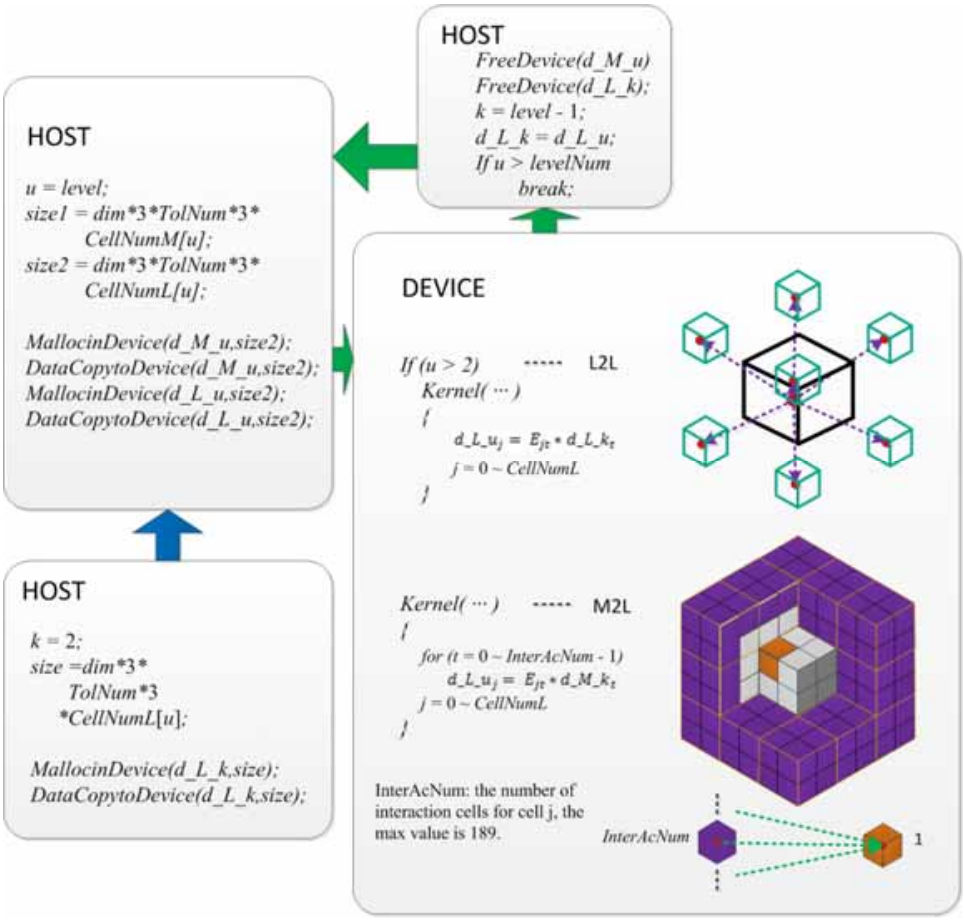


Figure 7: Global Hierarchy Substitution Transferring for L2L and M2L

L2L The processes of L2L is similar to M2M translation, except for the direction of translation is from parent *level* to child, as shown in Figure 7. It is worth noting that the performance of GHST is nice not only happening during two *level*'s *cells* such as L2L, but also between current *levelcells* such as M2L, FLE. As a result of

the highly reusable data in global memory of GPU, the speed up ratio of relational translations improves about 2 times relative to mapped memory method, as show inTable 5.

4.4 M2L

As one of the largest time-consumption part of FMM algorithm, M2L translation is studied specially by some researchers[Lexing; Biros; Zorin and Langston (2003; Lashuk; Chandramowlishwaran; Langston; Nguyen; Sampath; Shringarpure; Vuduc; Ying; Zorin and Biros (2009; Takahashi; Cecka; Fong and Darve (2012)]. By contrast with theory that apply translation stencil to reduce translations per *cell* from 189 to 119 mentioned in [Lashuk; Chandramowlishwaran; Langston; Nguyen; Sampath; Shringarpure; Vuduc; Ying; Zorin and Biros (2009)]by Lashuk, Takahashi proposed a more effectively method to reduce the reusable coefficient evaluation in M2L in[Takahashi; Cecka; Fong and Darve (2012)]. However, the total complexity actually increases from $512r'*r^2$ to $1728r'*r^2$ (r' and r is the order of series expansion) according his theory. In connection with this situation, we proposed a new solution method to decrease the complexity.

According Takahashi's theory, the parent *cell* contains the current *cell* is called *cluster*. The translation between source *cell* j and target *cell*i can be expressed as:

$$d\_L\_u_i = \int_{j=0}^{CellNumL} E_{ij} * d\_M\_k_j$$

Where E_{ij} is the translation operator, and d_L_u is the local expansion value of target *cell*i, and d_M_k is the moment value of source *cell* j.

As shown in Figure 2, there are at most 27 clusters near target cluster (contains itself), 27 *cells* near the target *cell* j, and 189 interaction source *cells* for target *cell*i. Thus, it request at most 189 translations for a target *cell*. From the Eq. (22)(23), we can find that the value is determined by the distance of center of *cell*i, j. Hence, the coefficient could be reused when the distance of center of *cell*i, j does not change. We should take coefficients of target *cell* of a *cluster* into matrix $[D_{cluster}]$, and the little square of matrix $[D_{cluster}]$ is an $r' * r$ (r' is the order of moment expansion, r is the order of local expansion) matrix $[D_{ij}]$ for *cell*i, j. The elements during matrix $[D_{ij}]$ are coefficient value between source *cell* j and target *cell*i that is evaluated by Eq. (21)(22).

Because of the same distance value between *cell*i, j, the $[D_{cluster}]$ just needs evaluating 27 types of interaction matrix $[D_{ij}]$, as shown in Figure 8, the different color square expresses different types of interaction matrix $[D_{ij}]$. The squares of $[D_{cluster}]$ in example shown in Figure 8 express that the coefficients between SC (source *cell* index) and TC (target *cell* index). The color squares transform into blank squares

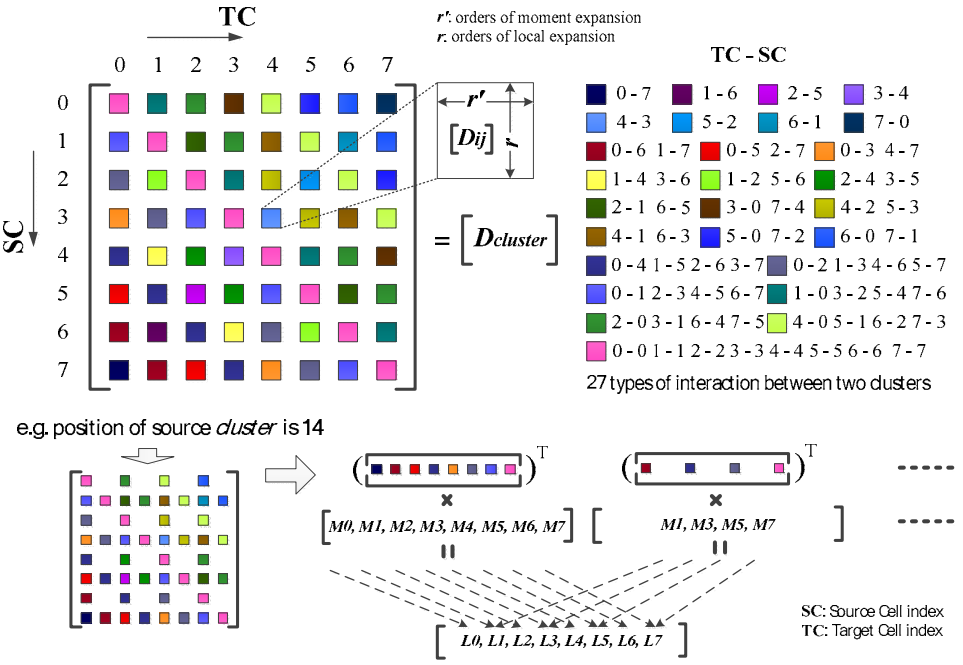


Figure 8: Optimization of M2L algorithm for parallel

means that the relationship between SC and TC are contiguous, and will not be calculated with M2L but NIC procedure. Table 6 has listed which square will change to blank (the color square is also numbered by 0˜26) as the source *cluster* appearing in different position relative to the target *cluster* (the index 0˜26 expresses different position as Figure 2 shows) during the computation. During the computation of M2L, MMP procedure need be applied. For improving the efficiency, this paper extracts the non-repetitive part from coefficient matrix and moment matrix to participate the multiplication, as shown in Figure 8. Compared the conventional method and optimization method proposed by Lashuk, the complexity of this method reduced to $117r'*r^2$.

4.5 Mixed parallel with Open-MP and Multi-core GPU

The most time-consumption calculations in FMM-DRBEM are MMP evaluation between [***H***] and [***W***], [***G***] and [***P***], respectively.In our algorithm, these procedures are dealt with multi-processors in CPU, and parallel translations of FMM-DRBEM in dual-core GeForce 590.

Because of the unequal time consumption for different MMP calculation, the im-

Table 6: The unavailable types of interaction with different source *cluster* index

*cluster*Index	Contiguous Relationship	*cluster*Index	Contiguous Relationship	*cluster*Index	Contiguous Relationship
0	(0)	9	(0, 4, 10)	18	(4)
1	(0, 1, 8)	10	(0, 1, 4, 5, 8, 10, 12, 16, 21)	19	(4, 5, 16)
2	(1)			20	(5)
3	(0, 2, 9)	11	(1, 5, 12)	21	(4, 6, 17)
4	(0, 1, 2, 3, 8, 9, 11, 13, 20)	12	(0, 2, 4, 6, 9, 10, 14, 17, 22)	22	(4, 5, 6, 7, 16, 17, 18, 19, 25)
5	(1, 3, 11)	13	all	23	(5, 7, 18)
6	(2)	14	(1, 3, 5, 7, 11, 12, 15, 18, 23)	24	(6)
7	(2, 3, 13)			25	(6, 7, 19)
8	(3)	15	(2, 6, 14)	26	(7)
		16	(2, 3, 6, 7, 13, 14, 15, 19, 24)	17	(3, 7, 15)

provement of mixed parallelism method could not achieve 100% for the efficiency. With the estimation approach proposed by Felipe A. Cruz et.al[Cruz; Knepley and Barba (2011)], the max improvement for these procedures is 80%.

5 Numerical Examples

This section demonstrates the accuracy of FMM-DRBEM algorithm and the efficiency of parallel code proposed in the above through two numerical examples. All the program codes are executed on a desktop computer with Intel Core 2 I7

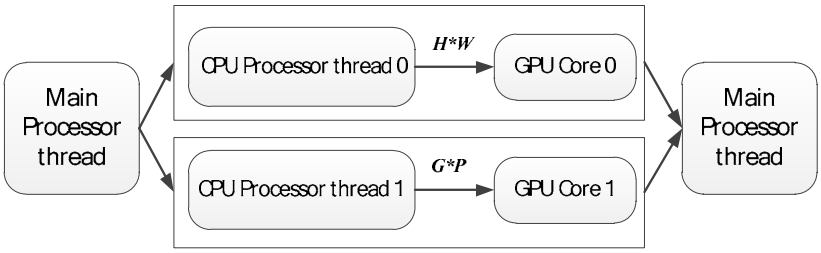


Figure 9: Mixed parallelize in FMM-DRBEM for MMP calculations

CPU, and the host memory is 8GB. The display card is NVIDIA GeForce 590 which enjoys two cores and 1.5GB space in each core. The peak floating-point arithmetic performance is 4.64 Tflops and 984 Gflops in single and double precision, respectively. In both two examples, the material parameters are: Shear modulus$\mu = 10^6 Pa$, Poisson's ratio$\nu = 0.3$,, and mass density $\rho = 7400 kg/m^3$.

5.1 Example 1

The first numerical example is to detect the efficiency of parallelize program code proposed in section 4. A cubic with $0.3m$ length in each edge is introduced, and the surface is faceted into numbers of elements through ACIS facet component developed by Spatial Company. This example lists the comparison in two different methods: conventional serial FMM-DRBEM algorithm and GPUparallel method. Thetime consumption of six translationsof$[\mathbf{G}]$ $[\mathbf{P}]$ in FMM-DRBEM algorithm is listed inTable 7, Table 8.

We should know that the expansion order not only affects the accuracy, but also the efficiency during the calculation. The two tables lists comparison of the time consumption in different expansion orders (*Nexp* = 3 and *Nexp* = 6). The effectiveness of *limit-number* in *leaf cell* is also considered in this example. Two different numbers of DOF are offered in this example: 3804 and 10122. The columns of every table represent the following quantities:

N: the number of DOFs in the body;

T: translation type;

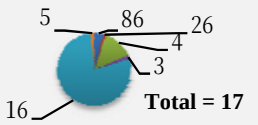
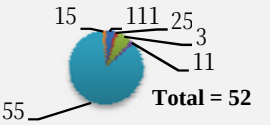
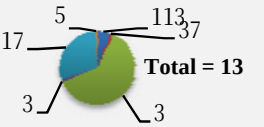
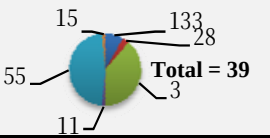
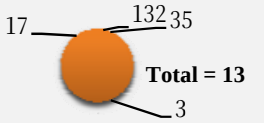
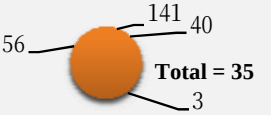
P: *limit-number*;

SG: singular GPU;

SR: speed-up ratio: CPU/SG;

Table 7 and Table 8 list result data of speed-up with GPU for translations in FMM

Table 7: The effectiveness of speed-up and memory requirement for translations in FMM with 3804 DOFs

N = 3804		*TolNum* = 10		*TolNum* = 28		SR
P	T	CPU(s)	SG(ms)	CPU(s)	SG(ms)	E2M FLE NIC M2M M2L L2L
20	E2M	60.088	696.83	167.89	1516.9	
	FLE	4.375	168.44	13.936	554.7	
	NIC	8.3361	2250.7	8.3026	2598.9	
	M2M	0.9922	295.18	7.1841	647.52	
	M2L	208.73	13163	1820.9	33109	
	L2L	1.2404	236.34	8.3164	550.62	
	Total	283.76	16807.5	2026.5	38978	
40	E2M	60.977	537.47	167.97	1344.6	
	FLE	4.4174	118.96	13.871	492.272	
	NIC	17.88	6160.2	17.796	6215.3	
	M2M	0.3416	100.829	2.4958	236.185	
	M2L	48.352	2909.2	421.12	7667.3	
	L2L	0.4176	76.047	2.8188	187.337	
	Total	132.39	9902.7	626.07	16143	
60	E2M	60.707	459.34	168.05	1189.6	
	FLE	4.454	99.352	13.89	464.72	
	NIC	19.774	6474.2	19.313	6534.9	
	M2M	0	0	0	0	
	M2L	27.063	1624.8	234.34	4194.1	
	L2L	0	0	0	0	
	Total	112.0	8657.7	435.6	12383	

with different number of DOFs. The data listed in the tables explain the detail time consumption during parallel and serial processes in different accuracy. The figures in these tables show the time consumption proportion of each translation procedure, and the speed-up ratio of each part. The max total speed-up ratio in Table 7 is 52

when *TolNum*= 28and *limit-number* = 60, and 32 in Table 8 when *TolNum*= 28and *limit-number* = 100.

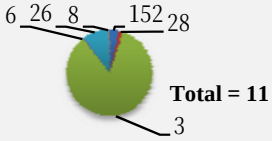
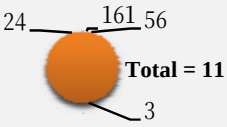
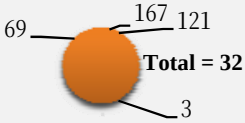
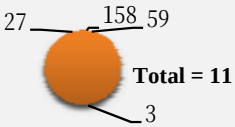
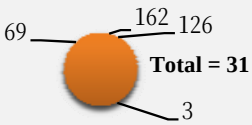
As these figures show, the total speed-up ratio is mostly determined by the speed-up ratio of the translations which occupy large proportion in total time consumption. The results will be affected by the number of *limit-number* in a *leaf*. Just like Table 7 shows, the total speed-up ratio is just 35 while *limit-number* = 60, far less than 52 while *limit-number* = 20. Because of unsuited *limit-number* respect to different problems, the low speed-up translation has large influence on total speed-up ratio. According from Table 7 and Table 8, it is obvious that the less of *limit-number*, the higher ratio for total speed-up. However, the total time is decreasing as the *limit-number* increase, except that the time consumption proportion of NIC procedure occupies too large.

In addition, we can find that the expansion order has deep influence for the effectiveness of speed-up. We list two kinds of *Nexp* in these tables, and both tables tell that the time consumption is increasing with *Nexp*, but the speed-up ratio is also improving obvious. That means the higher order of expansion applied, the higher effect can be achieved. For example, the total speed-up ratio is 31 which is almost three times than 11 obtained when *Nexp* = 10 in Table 8.

It is worth noting that the host memory collapse for the request memory of moment expansion (*TolNum*= 28) in all *level*s exceeds 8GB which is the limitation of our platform. Because of GHST method is applied in this paper, the scale of solvable problem in GPU is actually larger than CPU.

The GPU memory requirement in 10122 DOFs is listed in Figure 10. The Max requirement of memory is about 1400MB while *TolNum* equal to 28 (*Nexp* = 6). Although the GPU card we used has two independent cores and each has 1536 MB storage space, but the communication cost is too expensive to apply for large scale computation which needs interaction calculations time after time. The storage memory requirement will be affected by the effect of acceleration. If we increase the storage memory requirement, such as decreasing the *limit-number*and storing all intermediate data into GPU during computation, the speed-up effect will improve obviously, but the scale of solvable problem will decrease rapidly, the vice versa. Table 2 lists the estimate equation of memory in GPU, and Figure 10 lists the memory requirement during computation in 10122 DOFs. We can find that the max value is 1400MB which is close to the limit of the GPU card. Although this value could decrease by change code structure of parallelize program, the effectiveness will decrease rapidly and this kind of change is unnecessary for the limitation of CPU memory.

Table 8: The effectiveness of speed-up and memory requirement for translations in FMM with 10122 DOFs

N = 10122		*TolNum* = 10		*TolNum* = 28		SR
P	T	CPU(s)	SG(ms)	CPU(s)	SG(ms)	E2M FLE NIC M2M M2L L2L
60	E2M	887.51	5821	/	/	
	FLE	66.44	2362.8	/	/	
	NIC	424.55	133447	/	/	
	M2M	2.8232	514.28	/	/	
	M2L	414.01	15771	/	/	
	L2L	3.4717	422.11	/	/	
	Total	1798.8	158338	/	/	
100	E2M	889.89	5538.7	2540	15168	
	FLE	65.874	1166.1	220.48	1822.6	
	NIC	429.04	133905	470.59	138781	
	M2M	0	0	0	0	
	M2L	366.58	15079	3345.6	48583	
	L2L	0	0	0	0	
	Total	1751.4	155689	6576.7	204355	
140	E2M	864.59	5480	2444.5	15106	
	FLE	66.016	1123.9	211.39	1674.9	
	NIC	432.01	136140	435.36	140061	
	M2M	0	0	0	0	
	M2L	368.36	13730	3238.4	47116	
	L2L	0	0	0	0	
	Total	1731	156474	6329.7	224456	

5.2 Example 2

There will be two parts of analysis in this example. First, the effect of FMM-DRBEM is discussed through comparing the time consumption between classical BEM and FMM-DRBEM with different number of elements while computing

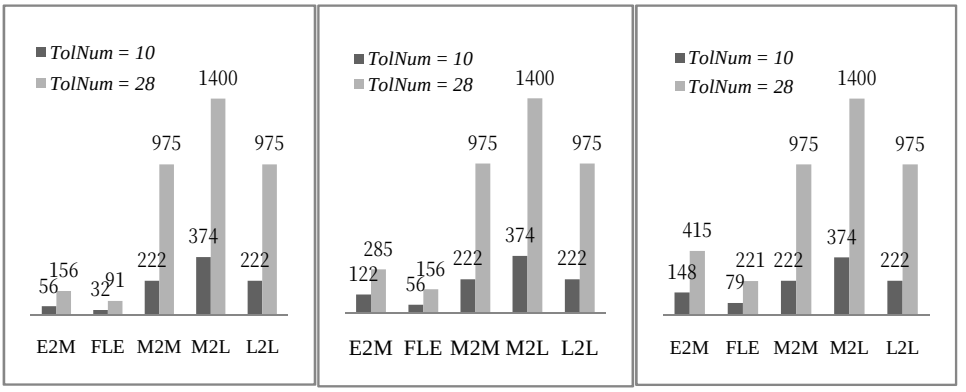


Figure 10: Memory requirement in GPU for each translation in different limit-number (**MB**)

MVP. Second, the accuracy of FMM-DRBEM proposed in this paper need to be demonstrated. The pedestal body is selected in this example, see Figure 11.

In the first part, the time consumption in different number of elements is summarized in Figure 12. In each kind number of elements, the time consumption in FMM-DRBEM is far less than classical BEM. We also find that the time cost of classical BEM will increase rapidly with the number of element, and the increase of FMM-DRBEM is far small, relatively. Because of the limitation of memory space, the large scale of problem can't be solved. So only three kind number of classical BEM data are listed in this figure.

In the second part, the natural frequencies are calculated with the algorithm proposed in section 2˜4. High accuracy is an important character of BEM with respect to FEM and FDM. However, due to the approximation of unknown $\mathbf{u}(\mathbf{x},t)$ (See Eq.(5)), the accuracy of DRBEM has been affected, but which could be fixed by introducing collocation points during computation[Chirino; Gallego; Sáez and Dominguez (1994)]. This paper lists the results corresponding to different number of collocation points to check the accuracy of this algorithm. The calculation results by FEM (Ansys and Hyperworks) are also exhibited for comparison. Through Figure 11(a), we find most orders of frequency are agreed with FEM except for the orders in 4, 5 have some distance with Ansys and Hyperworks. Then, the collocation points are applied for computation process (b, c in Figure 13). It is obvious that the agreement is closer than before, and the consistency is increasing as the number of collocation points grows.

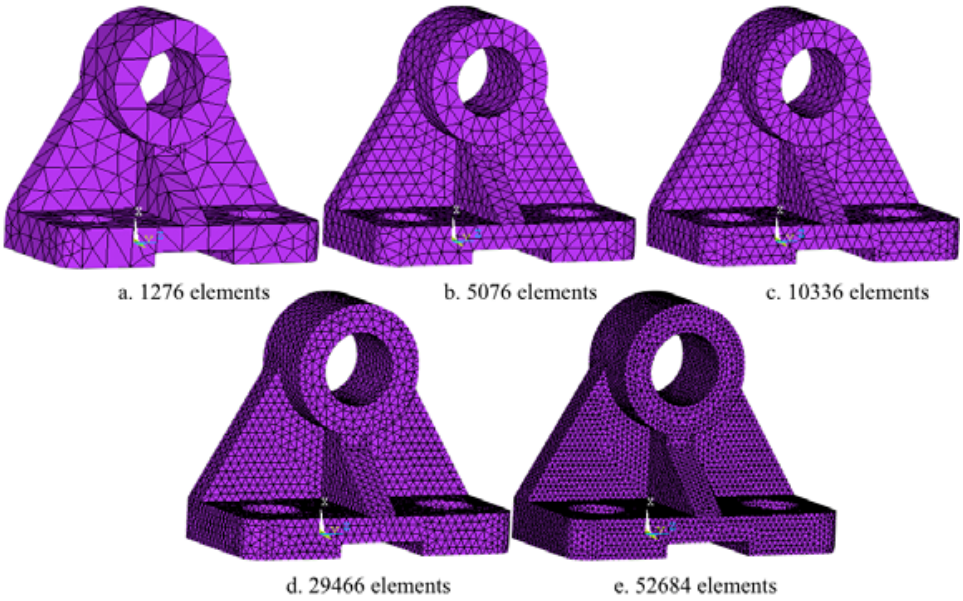


Figure 11: Pedestal bodywith different number of elements

Although the number of collocation points indeed has positive effect for the accuracy in general, but this kind of affect is not immovable. Actually, the accuracy of results evaluated with DRBEM is not bounded to the number of collocation points, the accuracy will slightly increases for a small increase of number of collocation points but decreases for further increase of them. The detail description of choosing collocation points refers to Chirino et.al [Chirino; Gallego; Sáez and Dominguez (1994)] and Agnantiaris et.al[Agnantiaris; Polyzos and Beskos (1996)], in which the view that a small number of collocation points improve the accuracy of the solution is proved.

6 Conclusion

Because of the complicated dynamic fundamental solutionsand low efficiency computation processes in classical BEM for 3D elastodynamic problems, this paper introduces a novel strategy with FMM-DRBEM and CUDA for accelerating computation process. First, the DRBEM is introduced to replace the dynamic fundamental solution with static ones which efficientlydecrease the complexity in computation. Then, FMM algorithmis introduced and modified to fit for MMP computation during DRBEM.Finally, we re-program the total computation codes into parallel codes

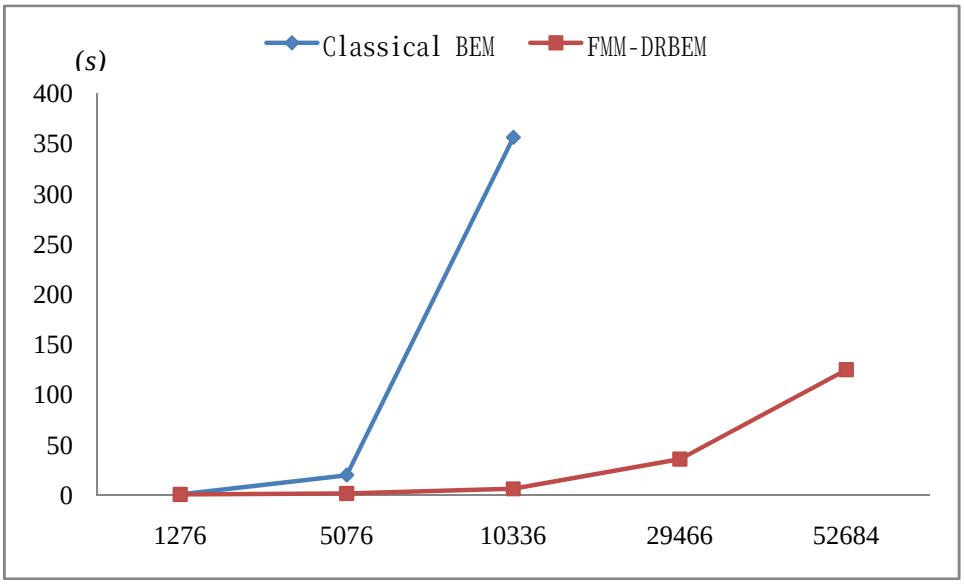
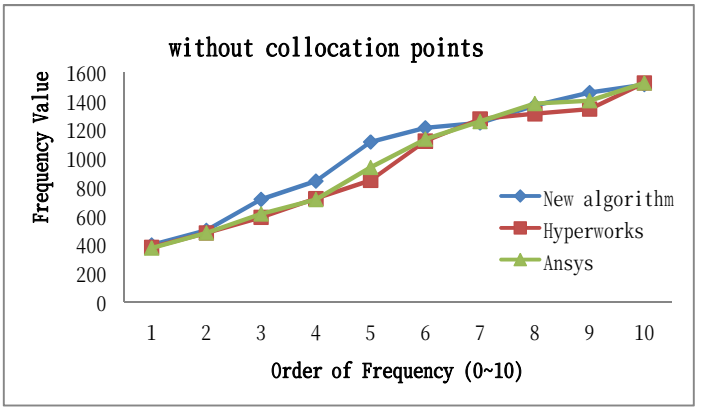


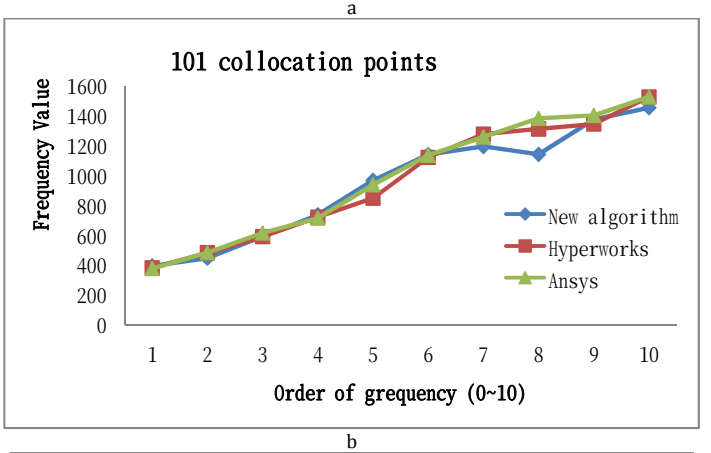
Figure 12: Time consumption in different methods

with CUDA to efficientlyaccelerate FMM computation processes. Apart from the general parallel strategies in GPU programming, we proposed some new strategies to optimize the computation processes such as PRS method for summation, GHST method for data transportation, and optimization of M2L procedure with respect to the characteristics of CUDA. In the last section, we propose numerical examples to examine the efficiency and accuracy of the algorithm proposed in this paper. The free vibration analysis for natural frequency is considered to check the accuracy through comparison with FEM, and the memory requirement in GPU is also checked to test the max magnitude could be solved in this platform. The affection of collocation points is considered, and the detail explanation for the itemcan be found in Chirino et.al[Chirino; Gallego; Sáez and Dominguez (1994)] and Agnantiaris et.al. [Agnantiaris; Polyzos and Beskos (1996)].

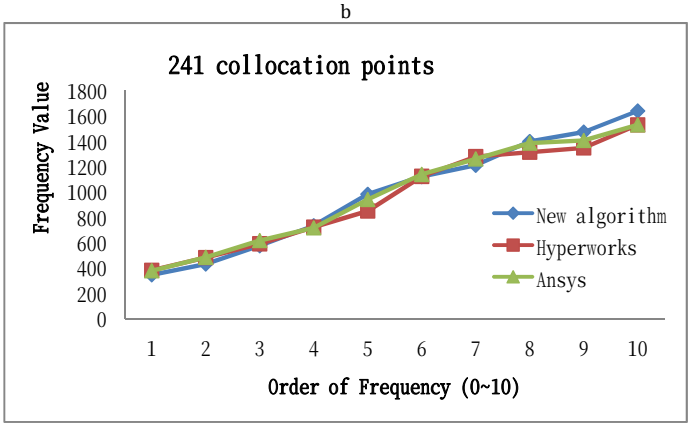
Acknowledgement: This research has been supported bythe National Natural Science Foundation of Chinaunder grant numbers 50975107, 51075162and 61173115.



a



b



c

Figure 13: Comparison of natural frequency results with FEM (Ansys and Hyperworks)

Appendix

$$U_{ij}(x,y) = \frac{1}{8\pi\mu} \int_{n=0}^{\infty}\int_{m=-n}^{n}\left[F_{ij,n,m}(x-y_c)\overline{R_{n,m}}(y-y_c) + G_{i,n,m}(x-y_c)(y-y_c)\overline{R_{n,m}}(y-y_c)\right], \tag{15}$$

$$T_{ij}(x,y) = E_{jklp}n_k(\mathbf{y})\frac{\partial}{\partial y_p}U_{il}(\mathbf{x},\mathbf{y}), \tag{16}$$

Where

$$F_{ij,n,m}(x-y_c) = \frac{\lambda+3\mu}{\lambda+2\mu}\delta_{ij}S_{n,m}(x-y_c) - \frac{\lambda+\mu}{\lambda+2\mu}(x-y_c)_j\frac{\partial}{\partial x_i}S_{n,m}(x-y_c), \tag{17}$$

$$G_{i,n,m}(x-y_c) = \frac{\lambda+\mu}{\lambda+2\mu}\frac{\partial}{\partial x_i}S_{n,m}(x-y_c), \tag{18}$$

Where

$$\begin{aligned} R_{n,\,m}(\mathbf{x}) &= \frac{1}{(n+m)!}P_n^m(\cos\theta)\,e^{im\phi}r^n, \\ S_{n,\,m}(\mathbf{x}) &= (n\,+\,m)!P_n^m(\cos\theta)\,e^{im\phi}\frac{1}{r^{n+1}}\,. \end{aligned} \tag{19}$$

The relationship between $R_{n,\,m}(\mathbf{x})$ and $S_{n,\,m}(\mathbf{x})$ shows in follow:

$$S_{n,\,m}(\mathbf{y},\mathbf{x}) = \int_{n'=0}^{\infty}\int_{m'=-n'}^{n'}\overline{R_{n',m'}(\mathbf{y})}S_{n+\,n',\,m+\,m'}(\mathbf{x}),$$

$$R_{n,\,m}(\mathbf{y},\mathbf{x}) = \int_{n'=0}^{n}\int_{m'=-n'}^{n'}R_{n',m'}(-\mathbf{y})\,S_{n-\,n',\,m-\,m'}(\mathbf{x}).$$

Moment Expansion:

$$M_{j,n,m}(y_c) = \int_{S_c}R_{n,m}(y-y_c)\,t_j(y)\,dS(y), \tag{20}$$

$$M_{n,m}(y_c) = \int_{S_c}(y-y_c)_jR_{n,m}(y-y_c)\,t_j(y)\,dS(y), \tag{21}$$

$$M_{j,n,m}\left(y_{c'}\right) = \int_{n'=0}^{n}\int_{m'=-n'}^{n'}R_{n',m'}\left(y_{c'}-y_c\right)M_{j,n-n',m-m'}(y_c), \tag{22}$$

$$M_{n,m}\left(y_{c'}\right) = \int_{n'=0}^{n} \int_{m'=-n'}^{n'} R_{n',m'}\left(y_{c'}-y_c\right)\left[M_{n-n',m-m'}\left(y_c\right)+\left(y_{c'}-y_c\right)_j M_{j,n-n',m-m'}\left(y_c\right)\right. . \tag{23}$$

Local Expansion

$$L_{j,n,m}\left(x_L\right) = (-1)^n \int_{n'=0}^{\infty} \int_{m'=-n'}^{n'} \overline{S_{n+n',m+m'}}\left(x_L-y_c\right) M_{j,n',m'}\left(y_c\right), \tag{24}$$

$$L_{n,m}\left(x_L\right) = (-1)^n \int_{n'=0}^{\infty} \int_{m'=-n'}^{n'} \overline{S_{n+n',m+m'}}\left(x_L-y_c\right)\left[M_{j,n',m'}\left(y_c\right)-\left(x_L-y_c\right)_j M_{j,n',m'}\left(y_c\right)\right], \tag{25}$$

$$L_{j,n,m}\left(x_{L'}\right) = (-1)^n \int_{n'=n}^{\infty} \int_{m'=-n'}^{n'} R_{n'-n,m'-m}\left(x_{L'}-x_L\right) L_{j,n',m'}\left(x_L\right), \tag{26}$$

$$L_{n,m}\left(x_{L'}\right) = (-1)^n \int_{n'=n}^{\infty} \int_{m'=-n'}^{n'} R_{n'-n,m'-m}\left(x_{L'}-x_L\right)\left[L_{n',m'}\left(x_L\right)-\left(x_{L'}-x_L\right)_j L_{j,n',m'}\left(x_L\right)\right]. \tag{27}$$

References

Agnantiaris, J. P.; Polyzos, D.; Beskos, D. E. (1996): Some studies on dual reciprocity BEM for elastodynamic analysis. *Computational Mechanics* 17(4): 270-277.

Agnantiaris, J. P.; Polyzos, D.; Beskos, D. E. (1998): Three-dimensional structural vibration analysis by the Dual Reciprocity BEM. *Computational Mechanics* 21(4): 372-381.

Agnantiaris, J. P.; Polyzos, D.; Beskos, D. E. (2001): Free vibration analysis of non-axisymmetric and axisymmetric structures by the dual reciprocity BEM. *Engineering Analysis with Boundary Elements* 25(9): 713-723.

Ahmad, S.; Banerjee, P. (1986): Free Vibration Analysis by BEM Using Particular Integrals. *Journal of Engineering Mechanics* 112(7): 682-695.

Benedetti, I.; Aliabadi, M. H. (2010): A fast hierarchical dual boundary element method for three-dimensional elastodynamic crack problems. *International Journal for Numerical Methods in Engineering* 84(9): 1038-1067.

Chaillat, S.; Bonnet, M.; Semblat, J.-F. (2007): A Fast Multipole Method formulation for 3D elastodynamics in the frequency domain. *Comptes Rendus MÃ©canique* 335(11): 714-719.

Chen, Y. H.; Chew, W. C.; Zeroug, S. (1997): Fast multipole method as an efficient solver for 2D elastic wave surface integral equations. *Computational Mechanics* 20(6): 495-506.

Chirino, F.; Gallego, R.; SÃ¡ez, A.; DomÃnguez, J. (1994): A comparative study of three boundary element approaches to transient dynamic crack problems. *Engineering Analysis with Boundary Elements* 13(1): 11-19.

Cruz, F. A.; Knepley, M. G.; Barba, L. A. (2011): PetFMM—A dynamically load-balancing parallel fast multipole library. *International Journal for Numerical Methods in Engineering* 85(4): 403-428.

Cruz, F. A.; Layton, S. K.; Barba, L. A. (2011): How to obtain efficient GPU kernels: An illustration using FMM & FGT algorithms. *Computer Physics Communications* 182(10): 2084-2098.

Epton MA, D. B. (1995): Multipole translation theory for the three-dimensional Laplace and Helmholtz equations. *SIAM J. Sci. Comput. (USA)*: 33.

Fujiwara, H. (1998): The fast multipole method for integral equations of seismic scattering problems. *Geophysical Journal International* 133(3): 773-782.

Fukui, T. (1998): Fast Multipole Boundary Element Method in 2D Elastodynamics. *Fukui Univ*: 8.

Greengard, L.; Rokhlin, V. (1987): A fast algorithm for particle simulations. *Journal of Computational Physics* 73(2): 325-348.

Greengard, L. F. (1988): The Rapid Evaluation of Potential Fields in Particle Systems. *The MIT Press*.

Gumerov, N. A.; Duraiswami, R. (2008): Fast multipole methods on graphics processors. *Journal of Computational Physics* 227(18): 8290-8313.

Hamada, T.; Narumi, T.; Yokota, R.; Yasuoka, K.; Nitadori, K.; Taiji, M. (2009): 42 TFlops hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence. Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. Portland, Oregon, *ACM*: 1-12.

Lashuk, I.; Chandramowlishwaran, A.; Langston, H.; Nguyen, T.-A.; Sampath, R.; Shringarpure, A.; Vuduc, R.; Ying, L.; Zorin, D.; Biros, G. (2009): A

massively parallel adaptive fast-multipole method on heterogeneous architectures. Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. Portland, Oregon, *ACM*: 1-12.

Lexing, Y.; Biros, G.; Zorin, D.; Langston, H. (2003): A New Parallel Kernel-Independent Fast Multipole Method. Supercomputing, 2003 ACM/IEEE Conference.

Liu, Y. J.; Mukherjee, S.; Nishimura, N.; Schanz, M.; Ye, W.; Sutradhar, A.; Pan, E.; Dumont, N. A.; Frangi, A.; Saez, A. (2011): Recent Advances and Emerging Applications of the Boundary Element Method. *Applied Mechanics Reviews* 64(3).

Nardini, D.; Brebbia, C. A. (1983): A new approach to free vibration analysis using boundary elements. *Applied Mathematical Modelling* 7(3): 157-162.

Nishimura, N. (2002): Fast multipole accelerated boundary integral equation methods. *Applied Mechanics Reviews* 55(4): 299-324.

NVIDIA. (2011): Nvidia CUDA Getting Started Guide For Microsoft Windows. from *http://www.nvidia.com/cuda*.

Rokhlin, V. (1985): Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics* 60(2): 187-207.

Rokhlin, V. (1990): Rapid solution of integral equations of scattering theory in two dimensions. *Journal of Computational Physics* 86(2): 414-439.

Takahashi, T.; Cecka, C.; Fong, W.; Darve, E. (2012): Optimizing the multipole-to-local operator in the fast multipole method for graphical processing units. *International Journal for Numerical Methods in Engineering* 89(1): 105-133.

Yan, Z. Y.; Zhang, J.; Ye, W. (2010): Rapid solution of 3-D oscillatory elastodynamics using the pFFT accelerated BEM. *Engineering Analysis with Boundary Elements* 34(11): 956-962.

Yoshida, K.-i.; Nishimura, N.; Kobayashi, S. (2001): Application of new fast multipole boundary integral equation method to crack problems in 3D. *Engineering Analysis with Boundary Elements* 25(4–5): 239-247.

Yoshida, K. (2001): Applications of Fast Multipole Method to Boundary Integral Equation Method. PHD, *Kyoto University*.