Copyright © 2012 Tech Science Press CMC, vol.28, no.1, pp.1-25, 2012

An $O(N)$ Fast Multipole Hybrid Boundary Node Method for 3D Elasticity

Q. Wang[1], **Y. Miao**[1,2], **H.P. Zhu**[1] **and C. Zhang**[3]

Abstract: The Hybrid boundary node method (Hybrid BNM) is a boundary type meshless method which based on the modified variational principle and the Moving Least Squares (MLS) approximation. Like the boundary element method (BEM), it has a dense and unsymmetrical system matrix and needs to be speeded up while solving large scale problems. This paper combines the fast multipole method (FMM) with Hybrid BNM for solving 3D elasticity problems. The formulations of the fast multipole Hybrid boundary node method (FM-HBNM) which based on spherical harmonic series are given. The computational cost is estimated and an $O(N)$ algorithm is obtained. The algorithm is implemented on a computer code written in C++. Numerical results demonstrate the accuracy and efficiency of the proposed technique.

Keywords: Hybrid boundary node method, Meshless method, Moving Least Squares approximation, Fast multipole method.

1 Introduction

In recent years, a new class of numerical methods, namely, the meshfree or meshless methods have been proposed and developed to solve the partial differential equations (PDEs). This is because the finite element method (FEM) or boundary element method (BEM) has much difficulty in solving problems involving changing domains such as large deformation or crack propagation. Besides, it may be arduous, time consuming and computationally expensive while meshing a 3D object with complicated geometry by FEM or BEM.

Many kinds of meshless methods have been proposed so far. The element free Galerkin method (EFG) [Belytschko, Lu and Gu (1994)] uses a global symmetric

[1] School of Civil Engineering and Mechanics, Huazhong University of Science and Technology, Wuhan 430074, China

[2] Corresponding Author. Tel: 86 27 87540172; Fax: 86 27 87542231; Y. Miao Email: my_miaoyu@163.com

[3] Institute of Rock and Soil Mechanics, Chinese Academy of Sciences, Wuhan 430071, China

weak form and the shape function comes from the moving least-squares (MLS) approximation [Belytschko, Krongauz, Organ, Fleming and Krysl (1996)]. In an attempt to avoid generation of the background cells, the meshless local Petrov-Galerkin (MLPG) approach [Atluri and Zhu (1998)] uses local form over local sub-domains. The EFG and MLPG are domain type meshless methods. Base on the idea of reducing the dimensionality of the problem like BEM, the boundary integral equations (BIEs) are combined with the meshless technique and boundary type meshless methods are developed, such as the boundary node method (BNM) [Mukherjee and Mukherjee (1997), Chati, Mukherjee and Mukherjee (1999)], the Galerkin boundary node method (GBNM) [Li and Zhu (2009)] and the boundary face method (BFM) [Zhang, Qin, Han and Li (2009)].

By combining the MLS approximation scheme with the hybrid displacement variational formula, a new boundary type meshless method, called Hybrid boundary node method (Hybrid BNM) [Zhang, Yao and Li (2002), Zhang, Tanaka and Matsumoto (2004), Zhang and Yao (2001), Zhang and Yao (2003), Zhang and Yao (2004)] is proposed. This method has been developed [Miao, Wang and Yu (2005), Miao and Wang (2006)] and applied to elastodynamics problems [Miao, Wang, Liao and Zheng (2009)], Helmholtz problems [Miao, Wang and Wang (2009)] and multi-domain problems [Wang, Zheng, Miao and Lv (2011)]. The Hybrid BNM not only reduces the spatial dimensions by one like BEM, but also does not require any cells neither for interpolation of the solution variables nor for the boundary integration. In fact, the Hybrid BNM requires only discrete nodes located on the surface of the domain and its parametric representation. However, as the traditional BEM, the Hybrid BNM has a dense and unsymmetrical system matrix, which requires $O(N^2)$ memory and $O(N^3)$ operations if a direct solver is used, such as Gaussian elimination method. The computational time of an iterative solver is still $O(M \times N^2)$, where M is the number of iterations, and the memory required is also $O(N^2)$. Therefore, the Hybrid BNM must be speeded up while dealing with large scale problems.

In the early 1980s several $O(N)$ and $O(N \log N)$ algorithms were introduced in N-body system for computing the interactions between the bodies, for which a standard solution leads to a computational complexity of $O(N^2)$. These algorithms were based on the expansion of potential field generated by N sources in multipolar or Taylor series and grouping far field influences. An oct-tree data structure [Barnes and Hut (1986)] is introduced to hierarchically subdivide the domain into well-separated areas, which can interact via the truncated expansions. The algorithm can reduce the computational complexity of the problem from $O(N^2)$ to $O(N \log N)$. By expanding the kernel in terms of spherical harmonic series and using the duality principle between the inner and outer expansions of harmonic functions, the concept of local expansion [Greengrad and Rokhlin (1987)] is introduced to translate and sum the effects of multiple remote multipole expansions into a single local value. This algorithm is known as the fast multipole method (FMM) and can further reduce the complexity of the problem to $O(N)$.

Because of the computational analogy between the interaction evaluation for the N-body problem and the matrix-vector multiplication, the FMM are widely employed to accelerate the solutions of PDEs through the BIEs in conjunction with iterative solvers. An original work of the application of the FMM for integral equations for two dimensional Laplace's equations [Rokhlin (1985)] shows the efficiency of the algorithm.

The implementation of FMM to accelerated BEM in 3D elasticity problems is investigated by many researchers and several $O(N)$ algorithms based upon spherical harmonic expansions are obtained. The original 3D elasticity fundamental solution can be decomposed into five terms and each of them can be expanded in terms of spherical harmonic series with their corresponding duality principle [Fu, Klimkowski and Rodin (1998)]. Starting from the kernel expansion of the fundamental solution of the Laplacian, a new multipole expansion together with the corresponding translations for the fundamental solution of linear elastostatics can be obtained [Yoshida, Nishimura and Kobayashi (2001)]. An $O(N)$ Taylor series multipole boundary element method for three-dimensional elasticity problems is presented [Popov and Power (2001)], in which an efficient clustering technique and an additional Taylor series expansion around the collocation points are used to further reduce the computational complexity to $O(N)$.

In this paper the Hybrid BNM is combined with FMM for solving 3D elasticity problems. The fast multipole boundary node method (FM-HBNM) for potential problems [Zhang, Tanaka and Endo (2005)] has been employed to study the thermal behavior of carbon nanotubes (CNT) composites [Zhang and Tanaka (2007b), Zhang and Tanaka (2008)], which is a very complex problem and almost impossible to obtain a reasonable discretization for the geometry with FEM. An original work of applying the FMM to accelerate Hybrid BNM for 3D elasticity problems has been investigated by the author's group [Wang, Miao and Zheng (2010)]. In this paper, the main formulations of the FM-HBNM for 3D elasticity problems which using the spherical harmonic series are given. The computational complexity is estimated and an $O(N)$ algorithm can be obtained.

This paper is organized as follows. The Hybrid BNM for 3D elasticity problems is reviewed in the second section. This is followed by the detail of the formulae of FM-HBNM for 3D elasticity problems. The procedures for the algorithm are summarized and the computational cost estimation is given next. Finally, in the fifth section, numerical results are given. The results show that the presented technique can further reduce the computational complexity to $O(N)$ while the accuracy remains high.

2 Review of the Hybrid BNM

In this section, the Hybrid BNM for 3D elasticity problems[Zhang and Yao (2004), Miao and Wang (2006)] is reviewed. The Hybrid BNM is based on a modified variational principle. In 3D elasticity, the functions in the modified variational principle that assumed to be independent are: displacements $\tilde{u}_i$ and tractions $\tilde{t}_i$ on the boundary and displacements u_i inside the domain. Consider a domain Ω enclosed by $\Gamma = \Gamma_u + \Gamma_t$ with $\bar{u}_i$ and $\bar{t}_i$ are the prescribed displacements and tractions, respectively. The corresponding variational functional is defined as follows:

$$\Pi_{HB=} \int_{\Omega} \frac{1}{2} u_{i,j} C_{ijkl} u_{k,l} \mathrm{d}\Omega - \int_{\Gamma} \tilde{t}_i (u_i - \tilde{u}_i) \mathrm{d}\Gamma - \int_{\Gamma_t} \bar{t}_i \tilde{u}_i \mathrm{d}\Gamma \tag{1}$$

where, the boundary displacements $\tilde{u}_i$ satisfies the essential boundary condition, i.e. $\tilde{u}_i = \bar{u}_i$ on Γ_u.

The displacements $\tilde{u}$ and tractions $\tilde{t}$ at the boundary are approximated by the MLS approximation as follows:

$$\tilde{u}(\mathbf{s}) = \sum_{J=1}^{N} \Phi_J(\mathbf{s}) \hat{u}_J \tag{2}$$

$$\tilde{t}(\mathbf{s}) = \sum_{J=1}^{N} \Phi_J(\mathbf{s}) \hat{t}_J \tag{3}$$

where N is the number of nodes for MLS approximation which located on the surface; $\hat{u}_J$ and $\hat{t}_J$ are nodal values, and $\Phi_J(\mathbf{s})$ is the shape function of the MLS approximation, corresponding to node $\mathbf{s}_J$.

The $\mathbf{u}$ and $\mathbf{t}$ inside the domain can be approximated by fundamental solutions as

$$\mathbf{u} = \left\{ \begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array} \right\} = \sum_{J=1}^{N} \left[\begin{array}{ccc} u_{11}^J & u_{12}^J & u_{13}^J \\ u_{21}^J & u_{22}^J & u_{23}^J \\ u_{31}^J & u_{32}^J & u_{33}^J \end{array} \right] \left\{ \begin{array}{c} x_1^J \\ x_2^J \\ x_3^J \end{array} \right\} \tag{4}$$

$$\mathbf{t} = \left\{ \begin{array}{c} t_1 \\ t_2 \\ t_3 \end{array} \right\} = \sum_{J=1}^{N} \left[\begin{array}{ccc} t_{11}^J & t_{12}^J & t_{13}^J \\ t_{21}^J & t_{22}^J & t_{23}^J \\ t_{31}^J & t_{32}^J & t_{33}^J \end{array} \right] \left\{ \begin{array}{c} x_1^J \\ x_2^J \\ x_3^J \end{array} \right\} \tag{5}$$

where $u_{ij}^J = u_{ij}(\mathbf{s}_J, Q)$ and $t_{ij}^J = t_{ij}(\mathbf{s}_J, Q)$ are the fundamental solutions; x_i^J are unknown parameters. For 3D elasticity problems, the fundamental solutions are

$$u_{ij}^J = \frac{-1}{16\pi r(1-v)\mu} \{(3-4v)\delta_{ij} - r_{,i} r_{,j}\} \tag{6}$$

$$t_{ij}^J = \frac{-1}{8\pi(1-v)r^2}\{[(1-2v)\delta_{ij}+3r_{,i}r_{,j}]\frac{\partial r}{\partial n}+(1-2v)(r_{,i}n_j-r_{,j}n_i)\} \tag{7}$$

where $r = r(\mathbf{s}_J, Q)$ and Q is the field point while $\mathbf{s}_J$ is the source point.

As the modified variational principle holds both in the whole domain and any sub-domain, the local sub-domain around each node can be taken into consideration. By taking variations in Eq. 1 with respect to the independent variables, the following set of equations, expressed in matrix form, are given as

$$\mathbf{Ux} = \mathbf{H\hat{u}} \tag{8}$$

$$\mathbf{Tx} = \mathbf{H\hat{t}} \tag{9}$$

where

$$U_{IJ} = \int_{\Gamma_I} \begin{bmatrix} u_{11}^J & u_{12}^J & u_{13}^J \\ u_{21}^J & u_{22}^J & u_{23}^J \\ u_{31}^J & u_{32}^J & u_{33}^J \end{bmatrix} h_I(Q)\mathrm{d}\Gamma \tag{10}$$

$$T_{IJ} = \int_{\Gamma_I} \begin{bmatrix} t_{11}^J & t_{12}^J & t_{13}^J \\ t_{21}^J & t_{22}^J & t_{23}^J \\ t_{31}^J & t_{32}^J & t_{33}^J \end{bmatrix} h_I(Q)\mathrm{d}\Gamma \tag{11}$$

$$H_{IJ} = \int_{\Gamma_I} \begin{bmatrix} \Phi_J(\mathbf{s}) & 0 & 0 \\ 0 & \Phi_J(\mathbf{s}) & 0 \\ 0 & 0 & \Phi_J(\mathbf{s}) \end{bmatrix} h_I(Q)\mathrm{d}\Gamma \tag{12}$$

$$\mathbf{x}^T = [x_1^1, x_2^1, x_3^1, ..., x_1^N, x_2^N, x_3^N] \tag{13}$$

$$\mathbf{\hat{u}}^T = [\hat{u}_1^1, \hat{u}_2^1, \hat{u}_3^1, ..., \hat{u}_1^N, \hat{u}_2^N, \hat{u}_3^N] \tag{14}$$

$$\mathbf{\hat{t}}^T = [\hat{t}_1^1, \hat{t}_2^1, \hat{t}_3^1, ..., \hat{t}_1^N, \hat{t}_2^N, \hat{t}_3^N] \tag{15}$$

where $h_I(Q)$ is a weight function, Γ_I is a regularly shaped local region around node $\mathbf{s}_I$ in the parametric representation space of the boundary surface. Therefore, the integrals in Eq. 10, Eq. 11 and Eq. 12 can be computed without using boundary elements [Zhang and Yao (2004), Miao and Wang (2006)].

For a general problem, either $\tilde{u}_i$ or $\tilde{t}_i$ are known at each node on the boundary and by rearranging Eq. 8 and Eq. 9, a final algebraic equation in terms of $\mathbf{x}$ only can be obtained as below:

$$\mathbf{Ax} = \mathbf{d} \tag{16}$$

For the node $\mathbf{s}_I$, if $\tilde{u}_i$ is known, select the correspond row in **U** to **A**, otherwise, select the correspond row in **T** to **A**, and the corresponding term of **d** comes from the matrix-vector product of $\mathbf{H}\hat{\mathbf{u}}$ or $\mathbf{H}\hat{\mathbf{t}}$. Then the unknown vector **x** is obtained by solving the final algebraic equation. The nodal values $\hat{\mathbf{u}}$ and $\hat{\mathbf{t}}$ on the boundary can be computed by the back-substitution of **x** into Eq. 8 and Eq. 9, then use Eq. 2 and Eq. 3 the displacements and tractions on the boundary can be obtained. The displacements and tractions at interior points can be evaluated by the traditional boundary integral equations.

The coefficient matrix **A** is dense and nonsymmetrical. The solution of system in Eq. 16 requires $O(N^2)$ memory and $O(N^3)$ CPU time to solve if using a direct solver such as Gauss elimination. When use an iterative solver, the most time-consuming part of computation will be the matrix-vector multiplications in each iteration step and $O(N^2)$ CPU time will be required. This is why the conventional Hybrid BNM is inefficient for large-scale problems, despite it is a boundary type meshless method which is robust in the meshing stage.

3 The FM-HBNM

The fast multipole method can be employed to accelerate the Hybrid BNM for solving Eq. 16. The main idea of the FMM is to translate the node-to-node interactions to cell-to-cell interactions, where the cells can be constructed by a hierarchical oct-tree structure. Iterative solvers (such as GMRES [Saad and Schultz (1986)]) are used in the FMM, where matrix-vector multiplications are computed using fast multipole expansions. Using the FMM for Hybrid BNM, the computational time of a problem can be reduced to $O(N)$.

In this section, the formulations of FM-HBNM are summarized. Consider the formula below, it is the inner product between one row of the matrix **A** in Eq. 16 and an iteration vector $\mathbf{x}'$ corresponding to the solution vector **x**, which is given by either

$$\sum_{J=1}^{N} \int_{\Gamma_I} u_{ij}^J h_I(Q) x'_{Ji} \mathrm{d}\Gamma(Q) \tag{17}$$

or

$$\sum_{J=1}^{N} \int_{\Gamma_I} t_{ij}^J h_I(Q) x'_{Ji} \mathrm{d}\Gamma(Q) \tag{18}$$

The first formula is computed for convenience. Consider two cells C_a and C_b, which contain N_a nodes and N_b nodes, respectively. The computational cost of a standard algorithm for the mutual interactions between the two groups is of order $O(N_a \times N_b)$ (see Fig. 1). In the cell-to-cell strategy, the computational cost is reduced to $O(N_a + N_b)$ (see Fig. 2).

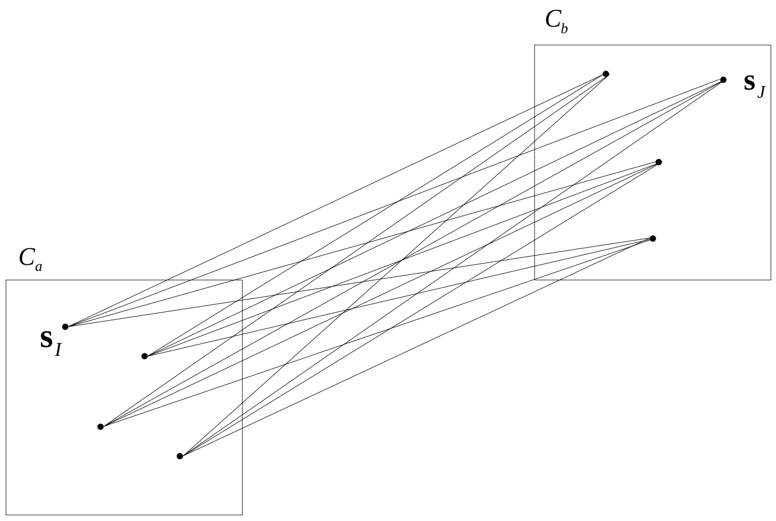

Figure 1: Node-to-node interactions

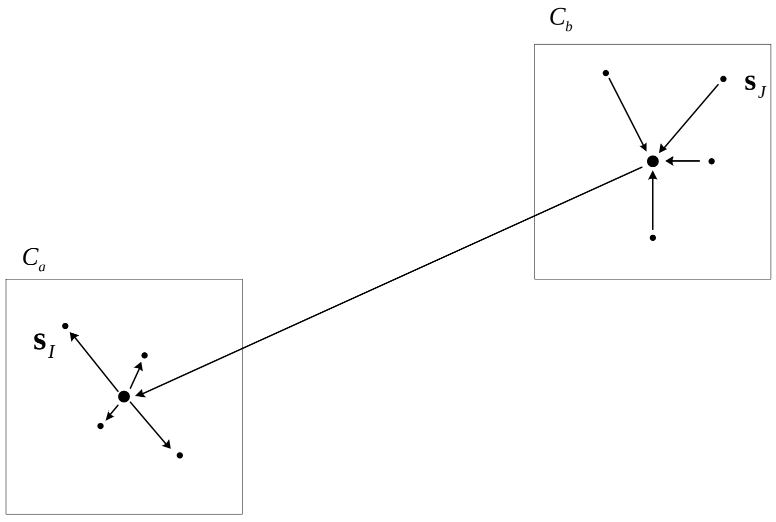

Figure 2: Cell-to-cell interactions

3.1 Multipole moments

The fundamental solution in Eq. 6 can be rewritten as the following form:

$$u_{ij}^J = u_{ij}^J(\mathbf{s}_J, Q) = \frac{1}{8\pi\mu}(\delta_{ij}\frac{2}{r(\mathbf{s}_J, Q)} - \frac{\lambda+\mu}{\lambda+2\mu}\frac{\partial}{\partial Q_i}\frac{\partial}{\partial Q_j} r(\mathbf{s}_J, Q)) \tag{19}$$

Rewrite Eq. 19 as

$$u_{ij}^{J}(\mathbf{s}_J, Q) = \frac{1}{8\pi\mu}(\delta_{ij}\frac{2}{r(\mathbf{s}_J, Q)} - \frac{\lambda+\mu}{\lambda+2\mu}\frac{\partial}{\partial Q_j}\frac{Q_i - \mathbf{s}_{Ji}}{r(\mathbf{s}_J, Q)}) \tag{20}$$

Since the condition $|O_1\mathbf{s}_J| < |O_1Q|$ holds (see Fig. 3), the following identity holds:

$$r(\mathbf{s}_J, Q) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\overline{S_{n,m}}(\overrightarrow{O_1Q})R_{n,m}(\overrightarrow{O_1\mathbf{s}_J}) \tag{21}$$

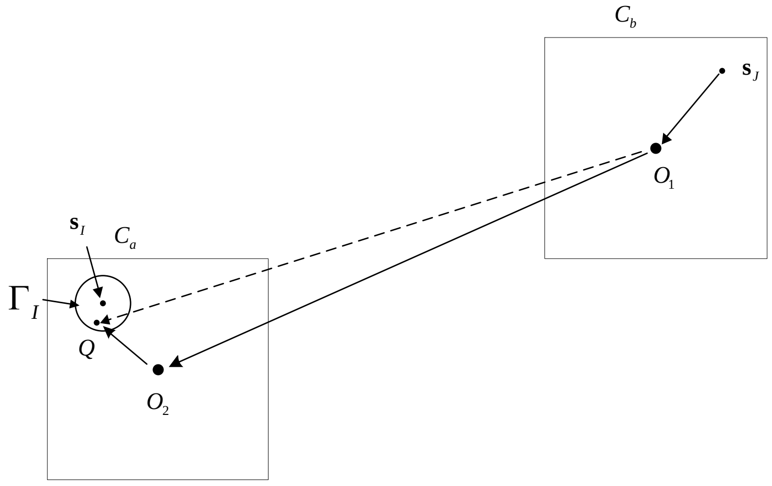

Figure 3: Interaction between two cells

By substituting Eq. 21 into Eq. 20, one can obtain

$$u_{ij}^{J} = \frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}(\overline{F_{ij,n,m}^{S}}(\overrightarrow{O_1Q})R_{n,m}(\overrightarrow{O_1\mathbf{s}_J}) + \overline{G_{j,n,m}^{S}}(\overrightarrow{O_1Q})(\overrightarrow{O_1\mathbf{s}_J})_i R_{n,m}(\overrightarrow{O_1\mathbf{s}_J})) \tag{22}$$

where

$$F_{ij,n,m}^{S}(\overrightarrow{Ox}) = \frac{\lambda+3\mu}{\lambda+2\mu}\delta_{ij}S_{n,m}(\overrightarrow{Ox}) - \frac{\lambda+\mu}{\lambda+2\mu}(\overrightarrow{Ox})_i\frac{\partial}{\partial x_j}S_{n,m}(\overrightarrow{Ox}) \tag{23}$$

$$G_{j,n,m}^{S}(\overrightarrow{Ox}) = \frac{\lambda+\mu}{\lambda+2\mu}\frac{\partial}{\partial x_j}S_{n,m}(\overrightarrow{Ox}) \tag{24}$$

$$R_{n,m}(\overrightarrow{Ox}) = \frac{1}{(n+m)!} P_n^m(\cos\theta) e^{im\phi} r^n \tag{25}$$

$$S_{n,m}(\overrightarrow{Ox}) = (n-m)! P_n^m(\cos\theta) e^{im\phi} \frac{1}{r^{n+1}} \tag{26}$$

(r, θ, ϕ) are the polar coordinates of the point x, P_n^m is the associated Legendre function and a superposed bar indicates the complex conjugate, respectively.

The solid harmonics $R_{n,m}$ and $S_{n,m}$ have the following properties:

$$R_{n,-m}(\overrightarrow{Ox}) = (-1)^m \overline{R_{n,m}}(\overrightarrow{Ox}) \tag{27}$$

$$S_{n,-m}(\overrightarrow{Ox}) = (-1)^m \overline{S_{n,m}}(\overrightarrow{Ox}) \tag{28}$$

Suppose that the boundary node $\mathbf{s}_I$ belongs to cell C_a and node $\mathbf{s}_J$ is one of the N_b nodes in cell C_b. Using Eq. 22, the sum in expression (17) for the nodes included in C_b can be obtained as

$$\begin{aligned} &\sum_{J=1}^{N_b} \int_{\Gamma_I} u_{ij}^J h_I(Q) x'_{Ji} \mathrm{d}\Gamma(Q) \\ &= \frac{1}{8\pi\mu} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \Big(\int_{\Gamma_I} \overline{F_{ij,n,m}^S}(\overrightarrow{O_1 Q}) h_I(Q) \mathrm{d}\Gamma(Q) M_{i,n,m}^1(O_1) \\ &+ \int_{\Gamma_I} \overline{G_{j,n,m}^S}(\overrightarrow{O_1 Q}) h_I(Q) \mathrm{d}\Gamma(Q) M_{i,n,m}^2(O_1) \Big) \end{aligned} \tag{29}$$

where $M_{i,n,m}^1(O_1)$ and $M_{i,n,m}^2(O_1)$ are multipole moments centered at O_1 (the center of C_b), expressed as

$$M_{i,n,m}^1(O_1) = \sum_{J=1}^{N_b} R_{n,m}(\overrightarrow{O_1 \mathbf{s}_J}) x'_{Ji} \tag{30}$$

$$M_{i,n,m}^2(O_1) = \sum_{J=1}^{N_b} (\overrightarrow{O_1 \mathbf{s}_J})_i R_{n,m}(\overrightarrow{O_1 \mathbf{s}_J}) x'_{Ji} \tag{31}$$

According to Eq. 27 and Eq. 28, the multipole moments have the following properties:

$$M_{i,n,-m}^1(O_1) = (-1)^m \overline{M_{i,n,m}^1(O_1)} \quad (m \geq 0) \tag{32}$$

$$M_{i,n,-m}^2(O_1) = (-1)^m \overline{M_{i,n,m}^2(O_1)} \quad (m \geq 0) \tag{33}$$

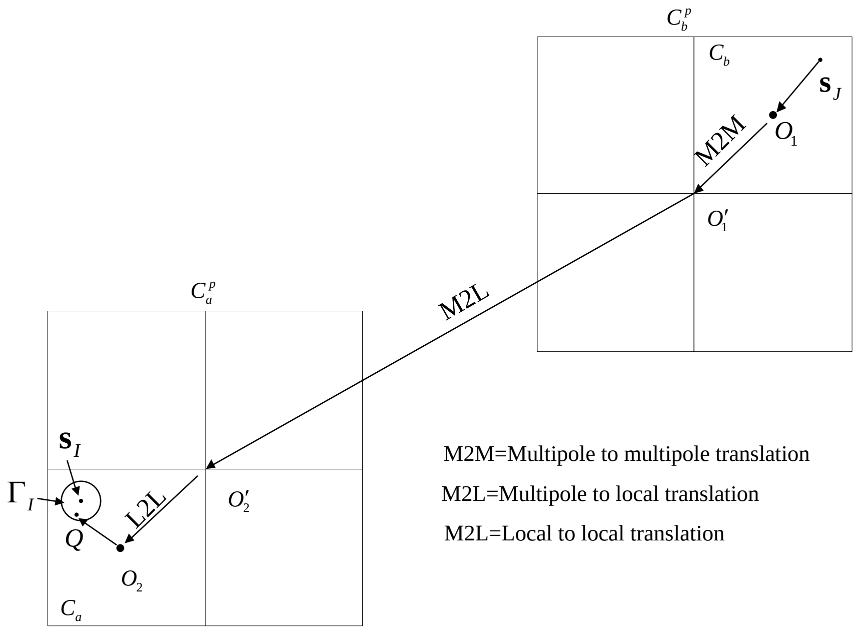
C_b^p
C_b
$\mathbf{s}_J$
M2M
O_1
O_1'
M2L
C_a^p
$\mathbf{s}_I$
Γ_I
L2L
O_2'
Q
O_2
C_a
M2M=Multipole to multipole translation
M2L=Multipole to local translation
M2L=Local to local translation

Figure 4: Conversions of multipole to local expansions

3.2 Multipole to multipole (M2M) translations

Suppose C_b is obtained by a larger cell C_b^p (see Fig. 4), the multipole moments about the center of C_b (O_1 in Fig. 4) can be translated to the center of C_b^p (O_1' in Fig. 4).

If $\overrightarrow{Ox}$ and $\overrightarrow{Oy}$ are two arbitrary vectors, then

$$R_{n,m}(\overrightarrow{yx}) = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{yO}) R_{n-n',m-m'}(\overrightarrow{Ox}) \tag{34}$$

Using Eq. 34, one can obtain the following two equations from Eq. 30 and Eq. 31:

$$M_{i,n,m}^{1}(O_1') = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O_1'O_1}) M_{i,n-n',m-m'}^{1}(O_1) \tag{35}$$

$$M_{i,n,m}^{2}(O_1') = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n',m'}(\overrightarrow{O_1'O_1}) (M_{i,n-n',m-m'}^{2}(O_1) - (\overrightarrow{O_1O_1'})_i M_{i,n-n',m-m'}^{1}(O_1)) \tag{36}$$

Eq. 35 and Eq. 36 are called as multipole to multipole (M2M) translations.

3.3 Multipole to local (M2L) translations

If $\overrightarrow{Ox}$ and $\overrightarrow{Oy}$ are two vectors such that $|\overrightarrow{Oy}| < |\overrightarrow{Ox}|$, then

$$S_{n,m}(\overrightarrow{yx}) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \overline{R_{n',m'}}(\overrightarrow{yO}) S_{n+n',m+m'}(\overrightarrow{Ox}) \tag{37}$$

Using Eq. 37, Eq. 29 can be rewritten as

$$\begin{aligned}
&\sum_{J=1}^{N_b} \int_{\Gamma_I} u_{ij}^J h_I(Q) x'_{Ji} \mathrm{d}\Gamma(Q) \\
&= \frac{1}{8\pi\mu} \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \int_{\Gamma_I} F_{ij,n,m}^R(\overrightarrow{O_2 Q}) h_I(Q) \mathrm{d}\Gamma(Q) L_{i,n',m'}^1(O_2) \\
&+ \frac{1}{8\pi\mu} \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \int_{\Gamma_I} G_{j,n,m}^R(\overrightarrow{O_2 Q}) h_I(Q) \mathrm{d}\Gamma(Q) L_{i,n',m'}^2(O_2)
\end{aligned} \tag{38}$$

where

$$L_{i,n',m'}^1(O_2) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (-1)^{n'} \overline{S_{n+n',m+m'}}(\overrightarrow{O_1 O_2}) M_{i,n,m}^1(O_1) \tag{39}$$

$$L_{i,n',m'}^2(O_2) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (-1)^{n'} \overline{S_{n+n',m+m'}}(\overrightarrow{O_1 O_2}) (M_{i,n,m}^2(O_1) - (\overrightarrow{O_1 O_2})_i M_{i,n,m}^1(O_1)) \tag{40}$$

Eq. 39 and Eq. 40 are known as the multipole to local (M2L) translations, which translate the multipole moments about the center of C_b to the local moments about the center of C_a (see Fig. 3).

Suppose C_a is obtained by a larger cell C_a^p (see Fig. 4). Assume that C_a^p and C_b^p are still far away from each other, from Eq. 39 and Eq. 40 we can obtain

$$L_{i,n',m'}^1(O'_2) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (-1)^{n'} \overline{S_{n+n',m+m'}}(\overrightarrow{O'_1 O'_2}) M_{i,n,m}^1(O'_1) \tag{41}$$

$$L_{i,n',m'}^2(O'_2) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (-1)^{n'} \overline{S_{n+n',m+m'}}(\overrightarrow{O'_1 O'_2}) (M_{i,n,m}^2(O'_1) - (\overrightarrow{O'_1 O'_2})_i M_{i,n,m}^1(O'_1)) \tag{42}$$

Eq. 41 and Eq. 42 translate the multipole moments about the center of C_b^p to the local moments about the center of C_a^p (see Fig. 4). Also, the local expansions have the following properties:

$$L^1_{i,n,-m}(O_2') = (-1)^m \overline{L^1_{i,n,m}(O_2')} \quad (m \geq 0) \tag{43}$$

$$L^2_{i,n,-m}(O_2') = (-1)^m \overline{L^2_{i,n,m}(O_2')} \quad (m \geq 0) \tag{44}$$

3.4 Local to local (L2L) translations

The center of the local moments will be shift from the center of C_a^p to C_a (see Fig. 4) by the following equations, which can be obtained from Eq. 34, Eq. 38, Eq. 43 and Eq. 44:

$$L^1_{i,n',m'}(O_2) = \sum_{n=n'}^{\infty} \sum_{m=-n}^{n} R_{n-n',m-m'}(\overrightarrow{O_2'O_2}) L^1_{i,n,m}(O_2') \tag{45}$$

$$L^2_{i,n',m'}(O_2) = \sum_{n=n'}^{\infty} \sum_{m=-n}^{n} R_{n-n',m-m'}(\overrightarrow{O_2'O_2})(L^2_{i,n,m}(O_2') - (\overrightarrow{O_2'O_2})_i L^1_{i,n,m}(O_2')) \tag{46}$$

Eq. 45 and Eq. 46 are called as local to local (L2L) translations. Eq. 38 can be rewritten as

$$\begin{aligned}
&\sum_{J=1}^{N_b} \int_{\Gamma_I} u^J_{ij} h_I(Q) x'_{Ji} \mathrm{d}\Gamma(Q) \\
&= \frac{1}{8\pi\mu} \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} A^1_{ij,n',m'} L^1_{i,n',m'}(O_2) + \frac{1}{8\pi\mu} \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} A^2_{j,n',m'} L^2_{i,n',m'}(O_2)
\end{aligned} \tag{47}$$

where

$$A^1_{ij,n',m'} = \int_{\Gamma_I} F^R_{ij,n',m'}(\overrightarrow{O_2Q}) h_I(Q) \mathrm{d}\Gamma(Q) \tag{48}$$

$$A^2_{j,n',m'} = \int_{\Gamma_I} G^R_{j,n',m'}(\overrightarrow{O_2Q}) h_I(Q) \mathrm{d}\Gamma(Q) \tag{49}$$

3.5 Expansions for expression (18)

The process for computing the expression (18) is exactly the same as the sum (17). Using the relation between t^J_{ij} and u^J_{ij}:

$$t^J_{ij}(\mathbf{s}_J, Q) = \frac{\partial}{\partial Q_l} u^J_{ik}(\mathbf{s}_J, Q) c_{klpj} n_p(Q) \tag{50}$$

where

$$c_{klpj} = \lambda \delta_{kl}\delta_{pj} + \mu(\delta_{kp}\delta_{lj} + \delta_{kj}\delta_{lp}) \tag{51}$$

one can obtain exactly the same formulations about the multipole moments and M2M, M2L and L2L translations, and Eq. 47 is replaced by

$$\begin{aligned}&\sum_{J=1}^{N_b}\int_{\Gamma_I} t_{ij}^J h_I(Q) x'_{Ji} \mathrm{d}\Gamma(Q)\\ &= \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} B^1_{ij,n',m'} L^1_{i,n',m'}(O_2) + \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} B^2_{j,n',m'} L^2_{i,n',m'}(O_2)\end{aligned} \tag{52}$$

where

$$B^1_{ij,n',m'} = \int_{\Gamma_I} \frac{\partial}{\partial Q_l} F^R_{ik,n',m'}(\overrightarrow{O_2Q}) c_{klpj} n_p(Q) h_I(Q) \mathrm{d}\Gamma(Q) \tag{53}$$

$$B^2_{j,n',m'} = \int_{\Gamma_I} \frac{\partial}{\partial Q_l} G^R_{k,n',m'}(\overrightarrow{O_2Q}) c_{klpj} n_p(Q) h_I(Q) \mathrm{d}\Gamma(Q) \tag{54}$$

In practical computations, the sum in the infinite series (35), (36), (41), (42), (45), (46), (47) and (52) are truncated after p terms.

4 FM-HBNM algorithm and estimation of the computational cost

In this section, the detail of the procedures for the algorithm proposed is summarized and the computational cost of the algorithm is estimated. The restarted preconditioned GMRES is employed as the iterative equation solver. An adaptive version of the FMM with a hierarchy of boxes which refine the computational domain is used.

4.1 FM-HBNM algorithm

The main procedures in the FM-HBNM are as follows:

Step 1: Discretization: For a given problem, create nodes which disturbed on the boundary of the domain in the same manner as in the original Hybrid BNM.

Step 2: Construction of oct-tree structure: Consider a smallest cube that can contain the entire domain needed to compute. Use it as the root cell (which is considered as level 0). Then divide this root cell into eight equal smaller cubes, which can be called as cells of level 1. Continue dividing in this way, that is, the cells of level $l+1$ is obtained from level l by subdividing of a cell into eight equal cells. The eight cells at level $l+1$ obtained by subdivision of the box at level l are considered as its children. Stop the subdivision of a cell while the number of nodes included in the cell is smaller than a prescribed number. Delete the child cell if it contains no node. A childless cell we call it leaf. Two cells are said to be neighbors if they are at the same level and share at least a vertex (a box is a neighbor of itself). Two cells are said to be well separated if they are at the same level but not neighbors. Each box b has an interaction list, whose members are the children of the neighbors of b's parent which are well separated from box b. An oct-tree data structure with hierarchy of cells can be constructed by this procedure.

Step 3: Determination of matrixes associated with cells: Choose the desired multipole expansion order p. With each box that $l > 1$, associate matrixes which describe the multipole moments $M^1_{i,n,m}$ and $M^2_{i,n,m}$ about the box center. With each box that $l > 2$, associate matrixes which describe the multipole moments $L^1_{i,n,m}$ and $L^2_{i,n,m}$ about the box center. With each leaf associate matrices described $A^1_{ij,n',m'}$, $B^1_{ij,n',m'}$, $A^2_{j,n',m'}$ and $B^2_{j,n',m'}$. Associate matrices U_L and T_L to the leaf described the near-field coefficients of U and L which will be computed directly.

Step 4: Computational of the integral associated with leaves: For each leaf, compute U_L, T_L, $A^1_{ij,n',m'}$, $A^2_{j,n',m'}$, $B^1_{ij,n',m'}$ and $B^2_{j,n',m'}$ by Eq. 10, Eq. 11, Eq. 48, Eq. 49, Eq. 53 and Eq. 54, respectively.

Step 5: Computation of the multipole moments at leaves: For an iteration vector x'_{Ji} form multipole moments $M^1_{i,n,m}$ and $M^2_{i,n,m}$ about the center of each leaf from all the nodes included in that leaf by Eq. 30 and Eq. 31.

Step 6: Translation of the multipole moments: Consider a non-leaf box of level l. Compute multipole moments $M^1_{i,n,m}$ and $M^2_{i,n,m}$ about the center of each box at level l by merging multipole moments from its children using Eq. 35 and Eq. 36 (M2M in Fig. 4). This procedure is repeated for $l \geq 2$ tracing the tree structure of boxes obtained in step upward (decreasing l). Steps 5 and 6 are considered as *upward pass*.

Step 7: Computation of the coefficients of the local expansions: Consider boxes at level l from level 2 to the finest level. For each box a at level l, convert the multipole moments $M^1_{i,n,m}$ and $M^2_{i,n,m}$ of each box b in the interaction list of box a to the local expansion about the centre of box a, using Eq. 41 and Eq. 42 (M2L in Fig. 4).

If $l > 2$, then shift the local expansion of a's parent to itself, using Eq. 45 and Eq. 46 (L2L in Fig. 4).

Add these two local expansions together. Step 7 is considered as *downward pass*.

Step 8: Evaluation of the integral in sum (17) or (18): The contribution of the far field is computed by Eq. 47 and Eq. 52 while the contribution of the near field deduced by the nodes contained in the neighborhood of the leaf is computed directly. Add the two parts.

Step 9: Update: Update the candidate vector and go back to step 5.

4.2 Estimation of the computational cost

The computational cost for Steps 5-8 are estimated since in FM-HBNM Steps 5-8 are iterated. When estimate the computational cost, we assume that the number of total boundary nodes is N and the boundary nodes distributed uniformly, the max number of boundary nodes in a leaf is M. The multipole expansions are truncated after p terms. Then the number of leaves s is N/M, the depth of the oct-tree d is $\log_8 s$.

Before the estimation, we have the following assumptions:

In Eq. 30 and Eq. 31, the computational time of $R_{n,m}(\overrightarrow{O_1\mathbf{s}_J})x'_{Ji}$ and $(\overrightarrow{O_1\mathbf{s}_J})_i R_{n,m}(\overrightarrow{O_1\mathbf{s}_J})x'_{Ji}$ are supposed as:

$$T\left\{R_{n,m}(\overrightarrow{O_1\mathbf{s}_J})x'_{Ji}\right\} = a_1$$

$$T\left\{(\overrightarrow{O_1\mathbf{s}_J})_i R_{n,m}(\overrightarrow{O_1\mathbf{s}_J})x'_{Ji}\right\} = a_2$$

In Eq. 35 and Eq. 36:

$$T\left\{R_{n',m'}(\overrightarrow{O'O})M^1_{i,n-n',m-m'}(O)\right\} = a_3$$

$$T\left\{R_{n',m'}(\overrightarrow{O'O})(M^2_{i,n-n',m-m'}(O) - (\overrightarrow{OO'})_i M^1_{i,n-n',m-m'}(O))\right\} = a_4$$

In Eq. 41 and Eq. 42:

$$T\left\{(-1)^{n'}\overline{S_{n+n',m+m'}}(\overrightarrow{O_1O_2})M^1_{i,n,m}(O_1)\right\} = a_5$$

$$T\left\{(-1)^{n'}\overline{S_{n+n',m+m'}}(\overrightarrow{O_1O_2})(M^2_{i,n,m}(O_1) - (\overrightarrow{O_1O_2})_i M^1_{i,n,m}(O_1))\right\} = a_6$$

In Eq. 45 and Eq. 46:

$$T\left\{R_{n-n',m-m'}(\overrightarrow{O_2O_3})L^1_{i,n,m}(O_2)\right\} = a_7$$

$$T\left\{R_{n-n',m-m'}(\overrightarrow{O_2O_3})(L^2_{i,n,m}(O_1) - (\overrightarrow{O_2O_3})_i L^1_{i,n,m}(O_2))\right\} = a_8$$

In Eq. 47:

$$T\left\{A^1_{ij,n',m'}L^1_{i,n',m'}(O_2)\right\} = a_9$$

$T\left\{A^2_{j,n',m'}L^2_{i,n',m'}(O_2)\right\} = a_{10}$

In Eq. 17

$T\left\{U_{Lij}x'_{Ji}\right\} = a_{11}$

where U_{Lij} is one element of U_L.

Now we can estimate the computational cost as follows:

Cost of Step 5:

In Step 5, the multipole moments at leaves are computed. The number of leaves is s, the number of nodes in a leaf is M and the infinite expansions are truncated after p. Notice that the multipole moments M_i^1 and M_i^2 have six components totally. For every component, the number of multipole moments is $(p+1)^2$. Because of the relations in Eq. 32 and Eq. 33, only $(p+1)(p+2)/2$ multipole moments are stored and computed. Therefore the total computational time for computation of multipole moments at leaves is:

$$T_5 = 3Ms(p+1)(p+2)(a_1+a_2)/2 = 3N(p^2+3p+2)(a_1+a_2)/2$$

Cost of Step 6:

In Step 6, the multipole moments are translated upward. The number of M2M translations via Eq. 35 and Eq. 36 at level l is 8^l. The M2M translations are iterated from $l = d$ to $l = 3$ and the total number of M2M translations is

$$n_6 = \sum_{l=3}^{d} 8^l = 8(s-64)/7$$

For one multipole moment $M^1_{i,n,m}$ or $M^2_{i,n,m}$, the computational time is $(n+1)^2 a_3$ or $(n+1)^2 a_4$, respectively. There are $(p+1)(p+2)/2$ multipole moments for every component of M_i^1 or M_i^2 and the computational time for per M2M translation is:

$$T'_6 = \sum_{n=0}^{p} 3(n+1)^3(a_3+a_4) = 3(p^4+6p^3+13p^2+12p+4)(a_3+a_4)/4$$

Therefore the cost for M2M translations is

$$T_6 = n_6 T'_6 = 6(s-64)(p^4+6p^3+13p^2+12p+4)(a_3+a_4)/7$$

Cost of Step 7:

In Step 7, the coefficients of the local expansions are computed by M2L translations and L2L translations.

The number of M2L translations via Eq. 41 and Eq. 42 at level l is 189×8^l, where 189 is the max number of cells in the interaction list of one cell. The M2L translations are iterated from $l = 2$ to $l = d$ and the total number of M2L translations is

$$n_{7M} = \sum_{l=2}^{d} 189 \times 8^l = 189 \times 8(s-8)/7$$

For one multipole moment $L^1_{i,n,m}$ or $L^2_{i,n,m}$, the computational time via Eq. 41 or Eq. 42 is $(p+1)^2 a_5$ or $(p+1)^2 a_6$, respectively. Sine Eq. 43 and Eq. 44, there are $(p+1)(p+2)/2$ multipole moments for every component of L^1_i or L^2_i and the computational time for per M2L translation is:

$$T'_{7M} = \sum_{n=0}^{p} 3(n+1)(p+1)^2(a_5+a_6) = 3(p^4+5p^3+11p^2+7p+2)(a_5+a_6)/2$$

The total computational time for M2L translations can be obtained as:

$$T_{7M} = n_{7M} T'_{7M} = 189 \times 12(s-8)(p^4+5p^3+11p^2+7p+2)(a_5+a_6)/7$$

The number of L2L translations via Eq. 45 and Eq. 46 at level l is 8^l and they are iterated from $l = d$ to $l = 3$, then the total number of M2M translations is

$$n_{7L} = \sum_{l=3}^{d} 8^l = 8(s-64)/7$$

For one multipole moment $L^1_{i,n,m}$ or $L^2_{i,n,m}$, the computational time via Eq. 45 or Eq. 46 is $((p+1)^2-n^2)a_7$ or $((p+1)^2-n^2)a_8$, respectively. Sine Eq. 43 and Eq. 44, there are $(p+1)(p+2)/2$ multipole moments for every component of L^1_i or L^2_i and the computational time for per L2L translation is:

$$T'_{7L} = \sum_{n=0}^{p} 3((p+1)^2-n^2)(n+1)(a_7+a_8)$$
$$= 3(p^4+20p^3+57p^2+40p+2)(a_7+a_8)/4$$

The total computational time for L2L translations can be obtained as:

$$T_{7L} = n_{7L} T'_{7L} = 2(s-8)(3p^4+20p^3+57p^2+40p+2)(a_7+a_8)/7$$

Cost of Step 8:

In Step 8, the integral in sum (17) or (18) are computed.

The contribution of the far field is computed by Eq. 47 or Eq. 52, the number of total nodes is N, then the total cost for Eq. 47 or Eq. 52 is:

$$T_{8F} = 3N(p^2+3p+2)(a_9+a_{10})/2$$

The contribution of the near field is computed directly. The number of leaves is N/M and the max number of neighbors of a leaf is 27. Therefore, the cost for direct computation per leaf is $27M^2a_{11}\times 9 = 243M^2a_{11}$ and the total computational time for direct computation is

$$T_{8D} = 243M^2\times N/Ma_{11} = 243MNa_{11}$$

Hence the total computational time per iteration is

$$T_{Total} = T_5+T_6+T_{7M}+T_{7L}+T_{8F}+T_{8D}$$

One can indicate that the total computational complexity of FM-HBNM is nearly $O(N)$. Since $a_3+a_4\approx a_5+a_6\approx a_7+a_8$, the cost $T_{7M}\approx 387T_6\approx 387T_{7L}$ and the complexities of T_6, T_{7M} and T_{7L} are $O(Np^4)$. This means that the M2L translation is the bottleneck in the algorithm. In order to further reduce the cost of M2L, a new diagonal form, which can further reduce the M2L cost to $O(p^3)$ is proposed [Greengard and Rokhlin (1997)]. Other methods such as constructing new adaptive tree structures [Cheng, Greengard and Rokhlin (1999), Shen and Liu (2007), Bapat and Liu (2010), Zhang and Tanaka (2007a)] are also investigated to further reduce the computational cost. Applying these methods in our algorithm is an important subject of the future research.

5 Numerical results

The proposed techniques have been implemented in C++. For the purpose of error estimation, a formula is defined as

$$e = \frac{1}{|u|_{max}}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(u_i^{(e)}-u_i^{(n)})^2} \tag{55}$$

where $u_i^{(e)}$ and $u_i^{(n)}$ refer to the exact and numerical solutions respectively and $|u|_{max}$ is the maximum value of u over N nodes.

Three models shown in Fig. 5 are chosen to verify our method, which are geometries of three iterations in creating the Menger Sponge. The Menger Sponge was first described by Austrian mathematician Karl Menger. In mathematics, it is a fractal curve. The Menger Sponge is constructed by dividing a unit cube into an array of 27 smaller cubes of side one third, then removing the central cube and six cubes at the centre of each face. This procedure is repeated with each of the 20 remaining smaller cubes. At any point in the process, there will be 20^n cubes remaining where n is the number of iterations which have been carried out. The Menger Sponge is the limit of this construction process as n tends to infinity. Fig. 5 shows three iterations in creating the Menger Sponge.

The Menger Sponge has topological dimension only 1, which means that in many ways its behavior is much closer to that of curves and line drawings than to surfaces or solid volumes. At the same time, the Menger Sponge in fact contains a distorted copy of every other topologically 1-dimension set. It means that any curve, line, diagram or graph can be distorted to fit in the sponge without self intersection and mathematicians describe the Menger Sponge as a "universal curv".

In the three models, the dimensions of the original cube are set to be 270 and the coordinate systems are located at the centers of the models. The properties are given by: Young's modules $E = 1.0$ and Poisson's ratio $v = 0.25$. Linear displacement fields are prescribed on all the boundaries as:

$$u_x = \frac{2x+y+z}{2}, u_y = \frac{x+2y+z}{2}, u_z = \frac{x+y+2z}{2} \tag{56}$$

In the FM-HBNM, a restarted preconditioned GMRES(m) with m=25 is employed as the preconditioner being the inverse of the blocked diagonal matrix corresponding to the nodes in leaves. All the infinite expansions are truncated after $p = 10$ and the maximum number of boundary nodes in a leaf box is set to be 60. The iteration is terminated when the relative error is less than 10^{-5}. All the computations are performed on a PC with a 2.67 GHz CPU and 8.0 GB RAM.

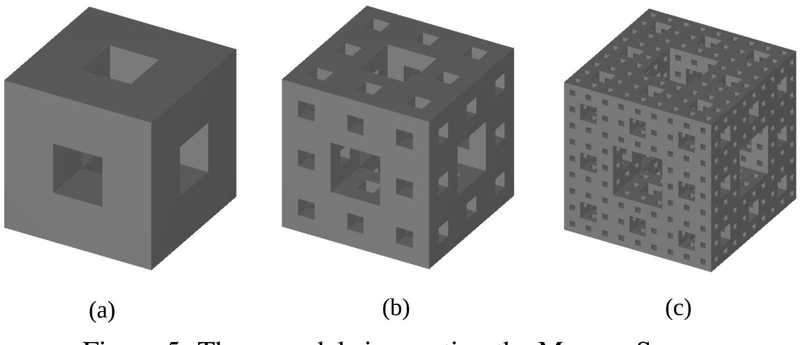

Figure 5: Three models in creating the Menger Sponge

Tab. 1-3 show the time results of the three models in Fig. 5. The first, second and third columns list the degrees of freedom (DOFs), number of levels used in the multipole hierarchy, and number of iterations of GMRES, respectively. The fourth, fifth and sixth columns indicate the time consumption of the upward pass, downward pass in FM-HBNM and the total CPU time of FM-HBNM, respectively. From Tab. 1-3 we can observe that the downward pass is the most time-consuming procedure in FM-HBNM, which verifies the estimation of the computational time in section 4. Figure 6 shows the CPU time per iteration of the FM-HBNM for the models (a) and (b) (see Fig. 5). In Fig. 6 one can observe that the CPU time of model (a) is less than model (b) while the DOFs are nearly equivalent. The main reason consists in the M2L translation, which is the bottleneck of the algorithm and the computational cost related with the number of cells in the interaction list. The details of the analysis for the computational cost can be found in section 4. Fig. 7 shows the relative errors of u_x for the model (a), which evaluated over 108 points uniformly distributed along the lines $x = y = 60$, $x = y = 110$, $x = y = 120$ and $x = y = 130$. The results demonstrate that the FM-HBNM is extremely effective for large-scale computation and the computational complex is nearly $O(N)$. It can also be concluded that FM-HBNM overcomes the restriction of the memory.

Table 1: Time results for model (a)

DOFs	Levels	Iterations	T-up	T-down	T-FMM
13 824	4	22	8	752	764
31 104	4	23	6	775	797
55 296	5	25	10	874	922
69 984	5	31	18	1 530	1 598
86 400	5	31	28	3 390	3 442
104 544	5	32	32	3 766	3 847
146 016	5	32	47	4 757	4 927
169 344	6	33	40	4 105	4 252
194 400	6	33	40	3 777	3 951
221 184	6	33	44	4 044	4 249
249 696	6	33	48	3 823	4 074
279 936	6	38	78	4 929	5 252
311 904	6	42	116	6 272	7 071

6 Conclusions

In this paper, an $O(N)$ fast multipole algorithm based upon the spherical harmonic series for 3D elasticity problems is proposed. Formulations for the FM-HBNM are

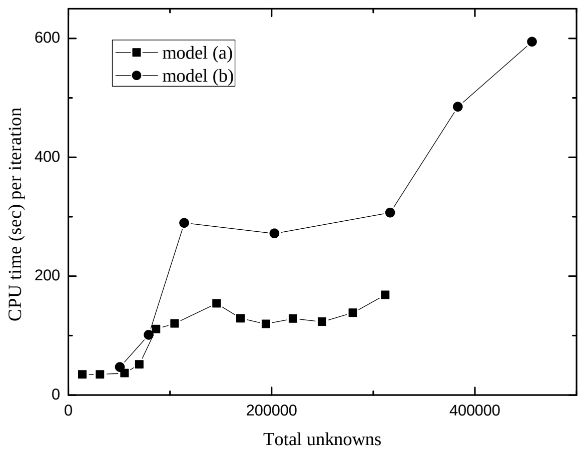

Figure 6: CPU time (sec) per iteration

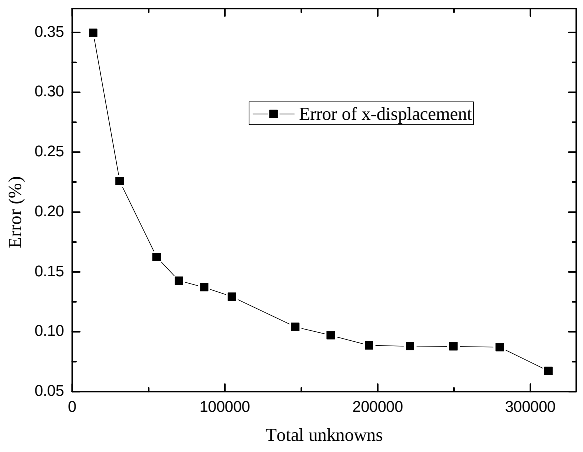

Figure 7: Error results for model (a)

Table 2: Time results for model (b)

DOFs	Levels	Iterations	T-up	T-down	T-FMM
50 688	5	37	16	1 667	1 733
79 200	5	47	38	4 574	4 747
114 048	5	44	65	12 558	12 738
202 752	5	47	71	12 494	12 770
316 800	6	45	104	12 809	13 800
383 328	6	62	213	28 504	30 064
45 6192	6	52	210	29 324	30 901

Table 3: Time results for model (c)

DOFs	Levels	Iterations	T-up	T-down	T-FMM
866 304	6	83	753	174 659	180 655
1 353 600	6	91	925	246 678	253 956

summarized and haven been implemented in a computer code written with C++. The computational cost is estimated and we can indicate that the M2L translation is most time-consuming in our algorithm. Numerical results show that the technique is effective and the computational time is nearly linear per iteration.

The FM-HBNM introduced in this paper retains the advantages of both the meshless method and the fast solver, which is especially applicable for large scale problems and problems with complicated geometries, such as porous materials, crack problems and composite materials.

From the fourth section and numerical results, one can see that the M2L translation is the bottleneck in FM-HBNM and there are still rooms to further reduce the computational time. Applying the new version of FMM and the new adaptive tree structures to our algorithm may be good choices. These are subjects of our future researches.

Acknowledgement: Financial support for the project from the National Basic Research Program of China (973 Program: 2011CB013800) and the Natural Science Foundation of China (No. 50808090).

References

Atluri, S. N.; Zhu, T. (1998): A new meshless local Petrov-Galerkin approach in computational mechanics. *Computational Mechanics*, vol. 22, pp. 117–127.

Bapat, M. S.; Liu, Y. (2010): A new adaptive algorithm for the fast multipole boundary element method. *CMES: Computer Modeling in Engineering & Science*, vol. 58, no. 2, pp. 161–183.

Barnes, J.; Hut, P. A. (1986): A hierarchical O(NlogN) force calculation algorithm. *Nature*, vol. 324, pp. 446–449.

Belytschko, T.; Krongauz, Y.; Organ, D.; Fleming, M.; Krysl, P. (1996): Meshless method: an overview and recent development. *Computer Methods in Applied Mechanics and Engineering*, vol. 139, pp. 3–47.

Belytschko, T.; Lu, Y. Y.; Gu, L. (1994): Element free Galerkin methods. *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 229–256.

Chati, M. K.; Mukherjee, S.; Mukherjee, Y. X. (1999): The boundary node method for three-dimensional linear elasticity. *International Journal for Numerical Methods in Engineering*, vol. 46, pp. 1163–1184.

Cheng, H.; Greengard, L.; Rokhlin, V. (1999): A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, vol. 155, pp. 468–498.

Fu, Y. H.; Klimkowski, K. J.; Rodin, G. J. (1998): A fat solution method for three dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, vol. 42, pp. 1215–1229.

Greengard, L.; Rokhlin, V. (1997): A new version of the fast multipole method for the laplace equation in three dimensions. *Acta Numerica*, vol. 6, pp. 229–269.

Greengrad, L.; Rokhlin, V. (1987): A fast algorithm for particles simulations. *Journal of Computational Physics*, vol. 73, pp. 325–348.

Li, X. L.; Zhu, J. (2009): A Galerkin boundary node method and its convergence analysis. *Journal of Computational and Applied Mathematics*, vol. 230, pp. 314–328.

Miao, Y.; Wang, Q.; Liao, B. H.; Zheng, J. J. (2009): A Dual Hybrid Boundary Node Method for 2D elastodynamics Problems. *CMES: Computer Modeling in Engineering and Science*, vol. 53, no. 1, pp. 1–22.

Miao, Y.; Wang, Y.; Wang, Y. H. (2009): A meshless hybrid boundary node method for helmholtz problems. *Engineering Analysis with Boundary Element*, vol. 33, no. 2, pp. 120–127.

Miao, Y.; Wang, Y. H. (2006): Meshless analysis for three-dimensional elasticity with singular hybrid boundary node method. *Applied Mathematics and Mechanics*, vol. 27, no. 6, pp. 673–681.

Miao, Y.; Wang, Y. H.; Yu, F. (2005): Development of hybrid boundary node method in two-dimensional elasticity. *Engineering Analysis with Boundary Elements*, vol. 29, pp. 703–712.

Mukherjee, Y. X.; Mukherjee, S. (1997): The boundary node method for potential problems. *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 797–815.

Popov, V.; Power, H. (2001): An O(N) Taylor series multipole boundary element methods for three-dimensional elasticity problems. *Engineering Analysis with Boundary Elements*, vol. 25, pp. 7–18.

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

Saad, Y.; Schultz, M. H. (1986): GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear system. *SIAM Journal on Scientific and Statistical Computing*, vol. 7, pp. 856–869.

Shen, L.; Liu, Y. (2007): An adaptive fast multipole boundary element method for three-dimensional potential problems. *Computational Mechanics*, vol. 39, pp. 681–691.

Wang, Q.; Miao, Y.; Zheng, J. J. (2010): The hybrid boundary node method accelerated by fast multipole expansion technique for 3D elasticity. *CMES: Computer Modeling in Engineering & Science*, vol. 70, no. 2, pp. 123–151.

Wang, Q.; Zheng, J. J.; Miao, Y.; Lv, J. H. (2011): The multi-domain hybrid boundary node method for 3D elasticity. *Engineering Analysis with Boundary Elements*, vol. 35, no. 6, pp. 803–810.

Yoshida, K.; Nishimura, N.; Kobayashi, S. (2001): Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering*, vol. 50, pp. 525–547.

Zhang, J. M.; Qin, X. Y.; Han, X.; Li, G. Y. (2009): A boundary face method for potential problems in three dimensions. *International Journal for Numerical Methods in Engineering*, vol. 80, pp. 320–337.

Zhang, J. M.; Tanaka, M. (2007): Adaptive spatial decomposition in fast multipole method. *Journal of Computational Physics*, vol. 226, pp. 17–28.

Zhang, J. M.; Tanaka, M. (2007): Systematic study of thermal properties of CNT composites by the fast multipole hybrid boundary node method. *Engineering Analysis with Boundary Element*, vol. 31, pp. 388–401.

Zhang, J. M.; Tanaka, M. (2008): Fast HdBNM for large-scale thermal analysis of CNT-reinforced composites. *Computational Mechanics*, vol. 41, pp. 777–787.

Zhang, J. M.; Tanaka, M.; Endo, M. (2005): The hybrid boundary node method accelerated by fast multipole expansion technique for 3D potential problems. *International Journal for Numerical Methods in Engineering*, vol. 63, pp. 660–680.

Zhang, J. M.; Tanaka, M.; Matsumoto, T. (2004): Meshless analysis of potential problems in three dimensions with the hybrid boundary node method. *International Journal for numerical methods in Engineering*, vol. 59, pp. 1147–1160.

Zhang, J. M.; Yao, Z. H. (2001): Meshless regular hybrid boundary node method. *CMES: Computer modeling in Engineering & Sciences*, vol. 2, pp. 307–318.

Zhang, J. M.; Yao, Z. H. (2003): The meshless regular hybrid boundary node method for 2-D linear elasticity. *Engineering Analysis with Boundary Element*, vol. 27, pp. 259–268.

Zhang, J. M.; Yao, Z. H. (2004): The regular hybrid boundary node method for three-dimensional linear elasticity. *Engineering Analysis with Boundary Element*, vol. 28, pp. 525–534.

Zhang, J. M.; Yao, Z. H.; Li, H. (2002): A hybrid boundary node method. *International Journal for Numerical Methods in Engineering*, vol. 53, pp. 751–763.