INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2005; **63**:660–680
Published online 3 March 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.1292

The hybrid boundary node method accelerated by fast multipole expansion technique for 3D potential problems

Jianming Zhang*,†, Masataka Tanaka and Morinobu Endo

Faculty of Engineering, Shinshu University, Nagano, 380-8553, Japan

SUMMARY

This paper presents a fast formulation of the hybrid boundary node method (Hybrid BNM) for solving problems governed by Laplace's equation in 3D. The preconditioned GMRES is employed for solving the resulting system of equations. At each iteration step of the GMRES, the matrix–vector multiplication is accelerated by the fast multipole method. Green's kernel function is expanded in terms of spherical harmonic series. An oct-tree data structure is used to hierarchically subdivide the computational domain into well-separated cells and to invoke the multipole expansion approximation. Formulations for the local and multipole expansions, and also conversion of multipole to local expansion are given. And a binary tree data structure is applied to accelerate the moving least square approximation on surfaces. All the formulations are implemented in a computer code written in C++. Numerical examples demonstrate the accuracy and efficiency of the proposed approach. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: meshless method; hybrid boundary node method; fast multipole method; moving least-squares approximation

1. INTRODUCTION

In numerical analysis of engineering problems two main difficulties arise. The first one is the discretization of the domain geometry, and the second the computational cost. Numerical models involve usually the finite element method (FEM) or the boundary element method (BEM) techniques. The FEM is a well-established and powerful method, in which the computational cost of solving the system of equations is proportional to the total number of degrees of freedom, while, a discretization of the domain of interest is required. This results in difficulties with remeshing in problems involving moving boundaries, large deformations or crack propagation. The task of mesh generation for complex geometries is often time-consuming and prone to errors. The BEM partially simplifies the discretization task, as it reduces the dimensionality of the problem by one. However, it results in a dense and unsymmetrical $N \times N$ system

*Correspondence to: J. Zhang, Faculty of Engineering, Shinshu University, Nagano 380-8553, Japan.
†E-mail: zhangjm@homer.shinshu-u.ac.jp

Received 3 June 2004
Revised 23 August 2004
Accepted 1 December 2004

of linear equations, where N is the total number of degrees of freedom. The computational cost and memory requirement for directly factoring such equation system increase with $O(N^3)$ and $O(N^2)$, respectively. This limits the BEM to relatively small scale problems.

In the last decade, a world-wide effort has been made to devise a new class of numerical methods, namely, the *meshfree* or *meshless* methods, aimed at eliminating the human-labour cost required for meshing the domains of complex-shape. Many types of meshless methods have been already proposed. These methods fall into two categories: the domain type and the boundary type. The domain type meshless methods are represented by the element free Galerkin method (EFG) [1], which uses a global symmetric weak form and the shape functions from the moving least-squares (MLS) approximation [2]. Although no mesh is required in the EFG method for the variable interpolation, background cells are inevitable for the 'energy' integration. Another domain type meshless method is the meshless local Petrov–Galerkin (MLPG) approach [3]. This method uses local weak forms over local sub-domains in an attempt to avoid generation of the background cells. Because no 'finite element/boundary element mesh' is required neither for the variable interpolation nor for the 'energy' integration, this method is truly meshless.

An example of the boundary type meshless methods is the boundary node method (BNM) [4], where the MLS approximation functions are inserted into the boundary integral equations (BIE). Although this method requires only a nodal data structure on the bounding surfaces of a body for approximation of boundary unknowns, it is not truly meshless, as the cell structure is again used for numerical integration. Another boundary type meshless method is the hybrid boundary node method (Hybrid BNM), proposed by Zhang *et al*. [5–10], which combines the MLS approximation scheme with the hybrid displacement variational formulation. The Hybrid BNM not only has the advantage of reducing the spatial dimensions by one as BEM or BNM, but also does not require any cells neither for interpolation of the solution variables nor for the boundary integration. Therefore, it is a truly meshless boundary-only method. In fact, the Hybrid BNM requires only discrete nodes located on the surface of the domain and its parametric representation. As the parametric representation of created geometry is used in most of CAD packages, it should be possible to exploit their Open Architecture features, and automatically obtain required coefficients (representation). However, like in the traditional BEM, its system matrix is dense and unsymmetrical, requiring $O(N^2)$ memory and $O(N^3)$ operations. Speeding up the Hybrid BNM with the fast multipole method (FMM) [11–14] is required for solving large scale problems. The FMM was introduced by Rokhlin [11] as a fast solution method for integral equations for two dimensional Laplace's equation and then developed by Greengard [12] as an algorithm for the rapid evaluation of potential and force fields in a large scale ensemble of charged particles. In Rokhlin's method, multipole moments are used to represent distant particle groups and a local expansion is introduced to evaluate the contribution from distant particles in the form of a series. The multipole moments associated with a distant group can be translated into the coefficients of the local expansion associated with a local group. In addition to Rokhlin's work, Greengard introduced a hierarchical decomposition of the domain geometry with a quad-tree in two dimensions and an oct-tree in three dimensions to carry out efficient and systematic grouping of particles. The FMM reduces the computational cost for the pair-wise force calculation from $O(N^2)$ to $O(N)$, hence making possible scientific and engineering computations of large scale problems.

The FMM has arisen in a variety of applications, ranging from computational astronomy to molecular dynamics and any problems related to the solution of Laplace's equation. Applying

FMM to accelerate BEM computation has been investigated by many researchers [15–17], and the computational costs of the fast BEM, including memory and CPU time, have been successfully reduced to $O(N)$. In this paper we combine the Hybrid BNM with FMM, and derive an efficient algorithm (here called FM–HBNM) not only in terms of computer costs but also in terms of human-labour costs, as mesh generation is not required.

However, unlike BEM using element-based interpolation functions to represent the approximate solutions of boundary variables, Hybrid BNM uses MLS approximation. The MLS approximation on a surface with a large number of nodes distributed can be costly and leads to a serious exhaustion of the computer memory, thus becomes another bottleneck for large-scale computation. In order to overcome this obstruct, we use a binary tree data structure, similar to the oct-tree data structure used in FMM, to speed up MLS approximation.

This paper is organized as follows. The Hybrid BNM is first outlined for problems in 3D potential theory. This is followed by coupling of the Hybrid BNM with the FMM. A new algorithm of the MLS approximation employing the binary tree data structure is described next. Finally, numerical results for sample problems are presented. Results from the Hybrid BNM and the combined FM–HBNM are compared with respect to accuracy and computational efficiency.

2. THE HYBRID BOUNDARY NODE METHOD

In this section we will give a description of Hybrid BNM, taking a 3D potential problem as an example (for detailed discussion see Reference [6]). The Hybrid BNM is based on a modified variational principle [18]. The functions in the modified variational principle assumed to be independent are: potential field within the domain, u, boundary potential field $\tilde{u}$ and boundary normal flux $\tilde{q}$. Consider a domain Ω enclosed by $\Gamma = \Gamma_u + \Gamma_q$ with prescribed potential $\bar{u}$ and normal flux $\bar{q}$ at the boundary portions Γ_u and Γ_q, respectively. The corresponding variational functional Π_{AB} is defined as

$$\Pi_{AB} = \int_\Omega \tfrac{1}{2} u_{,i} u_{,i} \, \mathrm{d}\Omega - \int_\Gamma \tilde{q}(u - \tilde{u}) \, \mathrm{d}\Gamma - \int_{\Gamma_q} \bar{q}\tilde{u} \, \mathrm{d}\Gamma \tag{1}$$

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

With the vanishing of $\delta\Pi_{AB}$ over the domain and its boundary, the following equivalent integral equations can be obtained:

$$\int_\Gamma (q - \tilde{q})\delta u \, \mathrm{d}\Gamma - \int_\Omega u_{,ii}\delta u \, \mathrm{d}\Omega = 0 \tag{2}$$

$$\int_\Gamma (u - \tilde{u})\delta\tilde{q} \, \mathrm{d}\Gamma = 0 \tag{3}$$

$$\int_{\Gamma_q} (\tilde{q} - \bar{q})\delta\tilde{u} \, \mathrm{d}\Gamma = 0 \tag{4}$$

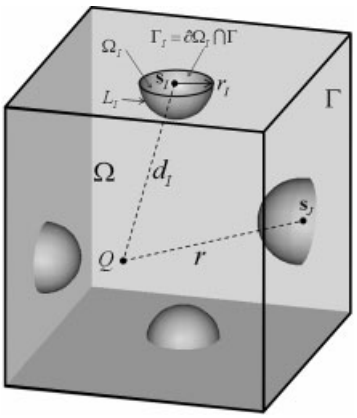

Figure 1. Local sub-domain centred at node $\mathbf{s}_I$ and the source point corresponding to node $\mathbf{s}_J$.

If we impose the flux boundary condition, $\tilde{q} = \bar{q}$, after the matrices have been computed, Equation (4) will be satisfied.

Equations (2) and (3) hold for any portion of the domain Ω, for example, a sub-domain Ω_I, defined as an intersection of the domain and a small sphere centred at node $\mathbf{s}_I$, and its boundary Γ_I and L_I (see Figure 1).

We use the following weak forms for the sub-domain and its boundary to replace Equations (2) and (3):

$$\int_{\Gamma_I + L_I} (q - \tilde{q}_s)\, v_I \, \mathrm{d}\Gamma - \int_{\Omega_I} u_{,ii} v_I \, \mathrm{d}\Omega = 0 \tag{5}$$

$$\int_{\Gamma_I + L_I} (u - \tilde{u}_s)\, v_I \, \mathrm{d}\Gamma = 0 \tag{6}$$

where v_I is a weight function; $\tilde{u}_s$ and $\tilde{q}_s$ are the boundary potential and normal flux at the boundary $\partial\Omega_I$, respectively.

We approximate $\tilde{u}$ and $\tilde{q}$ at the boundary Γ by the MLS approximation, as

$$\tilde{u}(\mathbf{s}) = \sum_{J=1}^{n} \Phi_J(\mathbf{s}) \hat{u}_J \tag{7}$$

$$\tilde{q}(\mathbf{s}) = \sum_{J=1}^{n} \Phi_J(\mathbf{s}) \hat{q}_J \tag{8}$$

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

$$\Phi_J(\mathbf{s}) = \sum_{j=1}^{m} p_j(\mathbf{s}) \left[A^{-1}(\mathbf{s}) B(\mathbf{s}) \right]_{jJ} \tag{9}$$

In the above equation, $p_j(\mathbf{s})$ provide a basis of order m consisting of monomials in (s_1, s_2). In this study, we take m to be 6, namely, $\mathbf{p}^{\mathrm{T}}(\mathbf{s}) = \left[1, s_1, s_2, s_1^2, s_1 s_2, s_2^2\right]$. Matrices $A(\mathbf{s})$ and $B(\mathbf{s})$ are defined by

$$A(\mathbf{s}) = \sum_{J=1}^{n} w_J(\mathbf{s})\mathbf{p}(\mathbf{s}_J)\,\mathbf{p}^{\mathrm{T}}(\mathbf{s}_J) \tag{10}$$

and

$$B(\mathbf{s}) = \left[w_1(\mathbf{s})\mathbf{p}(\mathbf{s}_1)\,, w_2(\mathbf{s})\mathbf{p}(\mathbf{s}_2)\,, \ldots, w_n(\mathbf{s})\mathbf{p}(\mathbf{s}_n)\right] \tag{11}$$

In Equations (10) and (11), $w_J(\mathbf{s})$ are weight functions. Gaussian weight function corresponding to node $\mathbf{s}_J$ can be written as

$$w_J(\mathbf{s}) = \begin{cases} \dfrac{\exp\left[-(d_J/c_J)^2\right] - \exp\left[-\left(\hat{d}_J/c_J\right)^2\right]}{1 - \exp\left[-\left(\hat{d}_J/c_J\right)^2\right]}, & 0 \leqslant d_J \leqslant \hat{d}_J \\ 0, & d_J \geqslant \hat{d}_J \end{cases} \tag{12}$$

where c_J is a constant controlling the shape of the weight function, and $\hat{d}_J$ is the size of the support for the weight function w_J. It can be seen from the above equation that the weight function has a compact support determined by parameter $\hat{d}_J$. The shape of the compact support is usually chosen to be circle in the meshless method literatures, while in this study, we choose ellipse for the shape of the compact support with $\hat{d}_J$ being the half-length of major axis of the ellipse. Denoting the half-length of minor axis by $\hat{d}'_J$, we have the following expression for d_J:

$$d_J = \sqrt{(s_1 - s_{J1})^2 + \frac{\hat{d}_J^2}{\hat{d}_J'^2}(s_2 - s_{J2})^2}$$

In order to ensure the regularity of $A(\mathbf{s})$, $\hat{d}_J$ and $\hat{d}'_J$ should be chosen in such a way that they are large enough to have a sufficient number of nodes which are covered in the domain of definition of every sample point. However, too large values of $\hat{d}_J$ and $\hat{d}'_J$ will lead to loss of the local character of the MLS approximation, or even an ill-conditioned matrix $A(\mathbf{s})$. In this study, $\hat{d}_J$ and $\hat{d}'_J$ is chosen such that $4m \sim 8m$ nodes are included in the support of a node. And the parameter c_J is taken to be such that $\hat{d}_J/c_J$ is constant and equals to 5.0. It should be noted that the MLS is an approximation rather than an interpolation, namely the MLS shape function does not possess the delta function property as the usual FEM or BEM shape function. We have constructed a generic MLS interpolation scheme on an arbitrary surface. For details see Reference [6].

In Equations (5) and (6), $\tilde{u}_s$ and $\tilde{q}_s$ at Γ_I can be represented by $\tilde{u}$ and $\tilde{q}$ expressed in Equations (7) and (8) since Γ_I is a portion of Γ, while $\tilde{u}_s$ and $\tilde{q}_s$ at L_I has not been defined yet. To solve this problem, we select v_I such that all integrals vanish over L_I. This can be easily accomplished by using the weight function in the MLS approximation for v_I, with the half-length of the major axis $\hat{d}_J$ of the support of the weight function being replaced by the

radius r_I of the sub-domain Ω_I, i.e.

$$v_I(Q) = \begin{cases} \dfrac{\exp\left[-(d_I/c_I)^2\right] - \exp\left[-(r_I/c_I)^2\right]}{1 - \exp\left[-(r_I/c_I)^2\right]}, & 0 \leqslant d_I \leqslant r_I \\ 0, & d_I \geqslant r_I \end{cases} \tag{13}$$

where d_I is the distance between a field point Q and the nodal point $\mathbf{s}_I$. With v_I vanishing at L_I, Equations (5) and (6) can be rewritten as

$$\int_{\Gamma_I} (q - \tilde{q}) v_I \, \mathrm{d}\Gamma - \int_{\Omega_I} u_{,ii} v_I \, \mathrm{d}\Omega = 0 \tag{14}$$

$$\int_{\Gamma_I} (u - \tilde{u}) v_I \, \mathrm{d}\Gamma = 0 \tag{15}$$

Making use of fundamental solutions, we approximate u inside the domain by

$$u = \sum_{J=1}^{N} u_J^s x_J \tag{16}$$

and hence at a boundary point, the normal flux is given by

$$q = \sum_{J=1}^{N} \frac{\partial u_J^s}{\partial n} x_J \tag{17}$$

where u_J^s is the fundamental solution; x_J are unknown parameters; N is the total number of boundary nodes; n is the outward normal vector to the boundary Γ. For 3D potential problems, the fundamental solution is written as

$$u_J^s = \frac{1}{4\pi} \frac{1}{r(Q, \mathbf{s}_J)} \tag{18}$$

where Q and $\mathbf{s}_J$ are field point and source point, respectively.

As u is expressed by Equation (16), the last integral on the left-hand side of Equation (14) vanishes if one excludes node $\mathbf{s}_I$, at which the singularity occurs, from the sub-domain Ω_I. This singularity will be taken into account when evaluating the boundary integrals. By substituting Equations (7), (8), (13), (16) and (17) into Equations (14) and (15), and omitting the vanishing terms, we have

$$\sum_{J=1}^{N} \int_{\Gamma_I} \frac{\partial u_J^s}{\partial n} v_I(Q) x_J \, \mathrm{d}\Gamma = \sum_{J=1}^{n} \int_{\Gamma_I} \Phi_J(\mathbf{s}) v_I(Q) \hat{q}_J \, \mathrm{d}\Gamma \tag{19}$$

$$\sum_{J=1}^{N} \int_{\Gamma_I} u_J^s v_I(Q) x_J \, \mathrm{d}\Gamma = \sum_{J=1}^{n} \int_{\Gamma_I} \Phi_J(\mathbf{s}) v_I(Q) \hat{u}_J \, \mathrm{d}\Gamma \tag{20}$$

Using the foregoing equations for all nodes, the following set of equations, expressed in matrix form, is given as

$$\mathbf{U}\mathbf{x} = \mathbf{H}\hat{\mathbf{q}} \tag{21}$$

$$\mathbf{V}\mathbf{x} = \mathbf{H}\hat{\mathbf{u}} \tag{22}$$

where

$$U_{IJ} = \int_{\Gamma_I} \frac{\partial u_J^s}{\partial n} v_I(Q)\,\mathrm{d}\Gamma \tag{23}$$

$$V_{IJ} = \int_{\Gamma_I} u_J^s v_I(Q)\,\mathrm{d}\Gamma \tag{24}$$

$$H_{IJ} = \int_{\Gamma_I} \Phi_J(\mathbf{s}) v_I(Q)\,\mathrm{d}\Gamma \tag{25}$$

$$\mathbf{x}^{\mathrm{T}} = [x_1, x_2, \ldots, x_N] \tag{26}$$

$$\hat{\mathbf{q}}^{\mathrm{T}} = \left[\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_N\right] \tag{27}$$

$$\hat{\mathbf{u}}^{\mathrm{T}} = \left[\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_N\right] \tag{28}$$

For a well-posed problem, either $\tilde{u}$ or $\tilde{q}$ is known at each node on the boundary. However, transformations between $\hat{u}_I$ and $\tilde{u}_I$, $\hat{q}_I$ and $\tilde{q}_I$ is necessary because the MLS approximation lacks the delta function property. For the panels where $\tilde{u}$ is prescribed, $\hat{u}_I$ is related to $\bar{u}_I$ by

$$\hat{u}_I = \sum_{J=1}^{N} R_{IJ}\tilde{u}_J = \sum_{J=1}^{N} R_{IJ}\bar{u}_J \tag{29}$$

and for the panels where $\tilde{q}$ is prescribed, $\hat{q}_I$ is related to $\bar{q}_I$ by

$$\hat{q}_I = \sum_{J=1}^{N} R_{IJ}\tilde{q}_J = \sum_{J=1}^{N} R_{IJ}\bar{q}_J \tag{30}$$

where $R_{IJ} = \left[\Phi_J\left(\mathbf{s}^I\right)\right]^{-1}$ (see Reference [19]).

Imposing the boundary condition, Equations (21) and (22) can be assembled into the following overall system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{d} \tag{31}$$

in which the Ith row of matrix $\mathbf{A}$ is supplied identically from that in $\mathbf{U}$ or $\mathbf{V}$ according to the boundary condition at the node $\mathbf{s}_I$, and the corresponding term of $\mathbf{d}$ comes from the matrix–vector product of $\mathbf{H}\hat{\mathbf{q}}$ or $\mathbf{H}\hat{\mathbf{u}}$.

The final matrix equation (31) is solved for unknown parameters **x**. Then, by back-substitution into Equations (21) and (22), the nodal values are obtained by

$$\hat{\mathbf{q}} = \mathbf{H}^{-1}\mathbf{U}\mathbf{x} \tag{32}$$

$$\hat{\mathbf{u}} = \mathbf{H}^{-1}\mathbf{V}\mathbf{x} \tag{33}$$

and potentials and normal fluxes at the boundary can be computed using Equations (7) and (8). Potentials and potential gradients at interior points are evaluated by the tradition boundary integral equations. Like other hybrid models (for example, the hybrid boundary element method [18]), the Hybrid BNM has a drawback of 'boundary layer effect' (the accuracy of the results in the vicinity of the boundary is very sensitive to the proximity of the interior points to the boundary). To avoid this drawback, an adaptive face integration scheme has been proposed in Reference [6]. As demonstrated, the Hybrid BNM is a boundary-only meshless approach. No boundary elements are used for both interpolation and integration purposes. Therefore, it can circumvent the discretization difficulty to a large extent.

3. ACCELERATING HYBRID BNM WITH FMM

The computational efficiency of the Hybrid BNM in comparison with 3D domain schemes, e.g. FEM or EFG, is similar to that of BEM. Actually, considering a 3D mesh with n^3 nodes, the computational cost for FEM is of order $O(n^3)$. On the other hand, the number of boundary nodes is around n^2, and hence, both the operation count and the memory requirements for the buildup of matrix equation (31) are of the order $O(n^4)$. The operation count increases to $O(n^6)$ if we attempt to solve the equation with conventional direct solvers such as Gaussian elimination. Therefore, although the dimensionality of the problem is reduced by one in Hybrid BNM, it is less computationally efficient than a domain scheme. However, the computational cost of the Hybrid BNM can be dramatically reduced to $O(n^2)$ when it is combined with the FMM [11–14]. In this section, we will explain in detail the implementation of FMM techniques in Hybrid BNM.

The FMM is called one of the top 10 algorithms of the 20th century. It is an algorithm for achieving fast products of particular dense matrices with vectors, and allows reduction of memory complexity. The FMM uses multipole expansions (in term of series) to approximate the effects of a distant group of particles (nodes in Hybrid BNM) on a local group, and thus achieve faster summation. In scientific computing we almost never seek exact answers. Instead of solving the problem we solve a 'nearby problem' that gives 'almost' the same answer. Moreover, FMM bounds the error analytically. We can determine how many terms are required in a multipole expansion to achieve a certain guaranteed level of accuracy. Therefore, FMM can be arbitrarily accurate. Another aspect of FMM is that it uses a hierarchical decomposition of space to define ever-larger groups as distances increase. For 3D problems, an oct-tree decomposition is usually employed. The details of the FMM algorithm can be found in References [13, 14].

In order to utilize the FMM techniques, it is necessary to use an iterative equation solver for solving Equation (31). The most time-consuming part of an iterative method, when employed for solving a linear system, is the calculation of matrix–vector product at each iteration step. In this paper, the restarted preconditioned GMRES [20] is employed, and an adaptive version

of FMM [14] used to accelerate the product. Following Reference [14], we first construct a hierarchy of boxes which refine the computational domain into smaller and smaller regions. At the refinement level 0, we have the entire computational domain. Refinement level $l+1$ is obtained recursively from level l by subdivision of each box into eight equal parts. This yields a natural tree structure, where the eight boxes at level $l+1$ obtained by subdivision of a box at level l are considered its children. Subdivision of the box is stopped when the number of nodes included in the box is smaller than a prescribed value. If a child box does not contain any node (is empty), we delete it. A childless box is called a leaf.

Definition 1
Two boxes are said to be *neighbours* if they are at the same level and share at least a vertex (a box is also a neighbour of itself).

Definition 2
Two boxes are said to be *well separated* if they are at the same level and are not neighbours.

Definition 3
With each box b we associate an *interaction list*, consisting of the children of the neighbours of b's parent which are well separated from box b.

Now consider the inner product between the Ith row of matrix $\mathbf{A}$ and an iteration vector $\mathbf{x}'$ corresponding to the solution vector $\mathbf{x}$, which is given by either

$$\sum_{J=1}^{N} \int_{\Gamma_J} u_J^s v_I(Q) x'_J \, \mathrm{d}\Gamma \tag{34}$$

or

$$\sum_{J=1}^{N} \int_{\Gamma_J} \frac{\partial u_J^s}{\partial n} v_I(Q) x'_J \, \mathrm{d}\Gamma \tag{35}$$

where x'_J is the Jth element of the iteration vector $\mathbf{x}'$.

For simplicity, only, we will ignore for a moment the sum in expression (35). Suppose the boundary node $\mathbf{s}_I$ belongs to a leaf of the hierarchical tree of boxes. We divide sum (34) into two parts. The first part is the sum of the contributions of the nodes contained in the neighbourhood of the leaf (these nodes are called *near nodes*), while the second is that of the contributions of the nodes that are outside the leaf's neighbourhood (these nodes are called *far nodes*). Sum (34) can then be expressed as

$$\sum_{J=1}^{N} \int_{\Gamma_J} u_J^s v_I(Q) x'_J \, \mathrm{d}\Gamma = \sum_{J}^{N_{\text{near}}} \int_{\Gamma_J} u_J^s v_I(Q) x'_J \, \mathrm{d}\Gamma + \sum_{J}^{N_{\text{far}}} \int_{\Gamma_J} u_J^s v_I(Q) x'_J \, \mathrm{d}\Gamma \tag{36}$$

where N_{near} and N_{far} are the numbers of the near nodes and far nodes, respectively.

We will compute the sum for the near nodes directly, while use multipole expansion to speed up the summation for the far nodes. Consider a leaf $\mathrm{B}_{\text{local}}$ and another leaf B_{far} which is on the interaction list of $\mathrm{B}_{\text{local}}$. $\mathrm{B}_{\text{local}}$ contains node $\mathbf{s}_I$, whilst B_{far} contains N_b nodes. We first create a spherical co-ordinate system with the origin located at the centre of B_{far}. Suppose the spherical co-ordinates of the quadrature point Q and node $\mathbf{s}_J$ are (r, θ, ϕ) and (ρ, α, β),

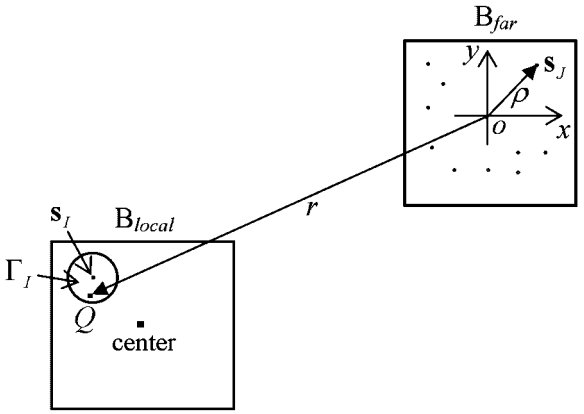

Figure 2. Co-ordinate system for multipole expansion.

respectively (see Figure 2). Since the condition $r>\rho$ holds, the fundamental solution (18) can be expanded in terms of spherical harmonic series [13, 14] as

$$u_J^s = \frac{1}{4\pi}\frac{1}{r\,(Q, \mathbf{s}_J)} = \frac{1}{4\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\rho^n Y_n^{-m}(\alpha, \beta)\frac{Y_n^m(\theta, \phi)}{r^{n+1}} \tag{37}$$

where $Y_n^m(x, y)$ is defined by

$$Y_n^m(x, y) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}\,P_n^{|m|}(\cos x)\mathrm{e}^{\mathrm{i}my} \tag{38}$$

with P_n^m expressed by Rodrigues' formula as

$$P_n^m(x) = (-1)^m\left(1-x^2\right)^{m/2}\frac{\mathrm{d}^m}{\mathrm{d}x^m}P_n(x)$$

and where $P_n(x)$ denotes the Legendre polynomial of degree n. Using Equation (37), the far field sum in Equation (36) for the nodes included in $\mathrm{B_{far}}$ can be expressed by

$$\sum_J^{N_b}\int_{\Gamma_I} u_J^s v_I(Q)x'_J\,\mathrm{d}\Gamma = \frac{1}{4\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}M_n^m\int_{\Gamma_I}\frac{Y_n^m(\theta, \phi)}{r^{n+1}}v_I(Q)\,\mathrm{d}\Gamma \tag{39}$$

where M_n^m are called multipole moments and given by

$$M_n^m = \sum_J^{N_b}\rho_J^n Y_n^{-m}\left(\alpha_J, \beta_J\right)x'_J \tag{40}$$

One might be inclined to think that the multipole expansion in Equation (39) makes the sum (34) more complicated and requires more operations to evaluate it. Actually, since the pair of points Q and $\mathbf{s}_J$ is separated in the expanded expression, the multipole moments M_n^m are related only to the nodes in $\mathrm{B_{far}}$ and the centre of $\mathrm{B_{far}}$. This allows us to compute M_n^m without concerning node $\mathbf{s}_I$. Furthermore, once the moments M_n^m are computed, they can be reused

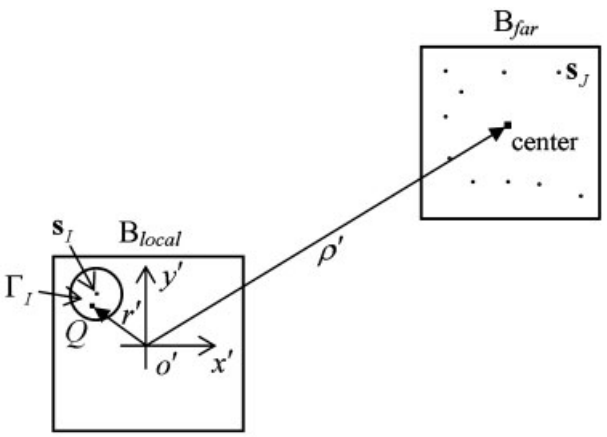

Figure 3. Co-ordinate system for local expansion.

for all the other corresponding nodes contained in boxes that are well separated from B_{far}. The re-use of computed terms is the key idea of the FMM, which is clarified in the following discussion.

In order to reuse terms in $\int_{\Gamma_I} Y_n^m(\theta,\phi)/r^{n+1} v_I(Q)\,\mathrm{d}\Gamma$, we move the origin of the spherical co-ordinate system to B_{local}'s centre, and denote the co-ordinates of B_{far}'s centre and point Q in the new co-ordinate system by (ρ',α',β') and (r',θ',ϕ'), respectively (see Figure 3). As B_{far} and B_{local} are well separated, the condition $\rho'>2r'$ is fulfilled. Hence $Y_n^m(\theta,\phi)/r^{n+1}$ can be again expanded in terms of spherical harmonics as [14]

$$\frac{Y_n^m(\theta,\phi)}{r^{n+1}} = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} \frac{i^{|k-m|-|k|-|m|} A_n^m A_j^k Y_{n+j}^{m-k}(\alpha',\beta')}{(-1)^n A_{n+j}^{m-k} \rho'^{j+n+1}} Y_j^k(\theta',\phi')\, r'^j \tag{41}$$

where A_n^m is defined by

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}$$

Substituting Equation (41) into Equation (39) yields

$$\sum_{J}^{N_b} \int_{\Gamma_I} u_J^s v_I(Q) x_J'\,\mathrm{d}\Gamma = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} L_j^k S_j^k \tag{42}$$

where L_j^k are called local moments and given by

$$L_j^k = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} M_n^m \frac{i^{|k-m|-|k|-|m|} A_n^m A_j^k Y_{n+j}^{m-k}(\alpha',\beta')}{(-1)^n A_{n+j}^{m-k} \rho'^{j+n+1}} \tag{43}$$

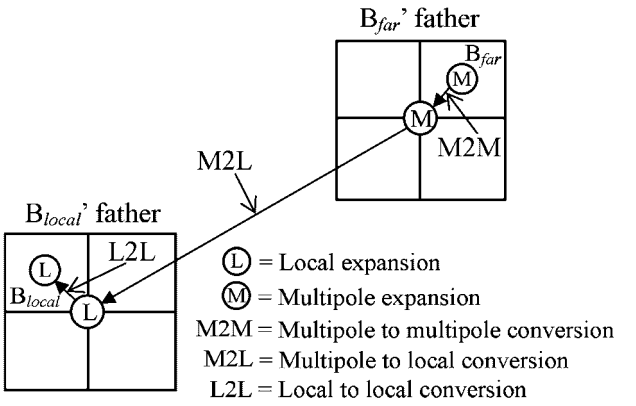

Figure 4. Conversions of multipole moments to local expansions.

and S_j^k given by

$$S_j^k = \frac{1}{4\pi} \int_{\Gamma_I} Y_j^k(\theta', \phi') r'^j v_I(Q) \, \mathrm{d}\Gamma \tag{44}$$

Equation (43) is usually called *multipole to local* translation, in which the coefficients are functions of the relative co-ordinates of the centre of B_{far} to the centre of B_{local}, only, and S_j^k involves only the local co-ordinates of the quadrature point Q about the centre of B_{local}. They can be computed independently.

We have so far discussed the evaluation of influences of a cluster of far nodes included in a leaf B_{far} on a node of another leaf B_{local}, where B_{far} is on B_{local}'s *interaction list*. If B_{far}'s parent belongs to the *interaction list* of B_{local}'s parent (see Figure 4), we first translate the multipole moments M_n^m about B_{far}'s centre into $M_j'^k$, which are about the centre of B_{far}'s parent, by [13, 14]

$$M_j'^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} M_{j-n}^{k-m} \frac{i^{|k|-|m|-|k-m|} A_n^m A_{j-n}^{k-m} \rho^n Y_n^{-m}(\alpha, \beta)}{A_j^k} \tag{45}$$

where (ρ, α, β) are the spherical co-ordinates of the centre of B_{far} with the origin of the co-ordinate system located at the centre of B_{far}'s parent.

Then, $M_j'^k$ into $L_n'^m$ is translated using Equation (43). Finally, we shift the centre of the local moments $L_n'^m$ from the centre of B_{local}'s parent to B_{local}'s employing the following equation [13, 14]:

$$L_j^k = \sum_{n=j}^{\infty} \sum_{m=-n}^{n} L_n'^m \frac{i^{|m|-|m-k|-|k|} A_{n-j}^{m-k} A_j^k Y_{n-j}^{m-k}(\alpha, \beta) \rho^{n-j}}{(-1)^{n+j} A_n^m} \tag{46}$$

where (ρ, α, β) are the spherical co-ordinates of the centre of B_{local} with the origin of the co-ordinate system located at the centre of B_{local}'s parent.

Equations (45) and (46) are called *multipole to multipole* and *local to local* translation, respectively. Together with the *multipole to local* translation (Equations (43)), these translations

are denoted by M2M, L2L and M2L in Figure 4, respectively. If B_{far}'s grandparent belongs to the *interaction list* of B_{local}'s grandparent, we first translate the multipole moments M_n^m from the centre of B_{far} to the centre of B_{far}'s parent, next from the centre of B_{far}'s parent to the centre of B_{far}'s grandparent, using Equation (45). Then, we convert M_n^m into L_j^k from the centre of B_{far}'s grandparent to the centre of B_{local}'s grandparent using Equation (43). Finally, we translate the local moments L_j^k from the centre of B_{local}'s grandparent to the centre of B_{local}'s parent, and subsequently to the centre of B_{local}, using Equation (46). In general, if one of B_{far}'s ancestors at level l belongs to the interaction list of one of B_{local}'s ancestors, the above process is repeated recursively until level l. In this scheme, all the *far to near* conversions are performed between a given box and boxes on its *interaction list*. This guarantees that condition $\rho' > 2r'$ holds as required by Equation (43) and maintains a uniform precision.

The computation process of sum (35) is the same as that of sum (34), except that Equation (42) is replaced by

$$\sum_{J}^{N_b} \int_{\Gamma_I} \frac{\partial u_J^s}{\partial n} v_I(Q) x_J' \, \mathrm{d}\Gamma = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k \frac{\partial S_j^k}{\partial n} \tag{47}$$

where $\partial S_j^k / \partial n$ is given by

$$\frac{\partial S_j^k}{\partial n} = \frac{1}{4\pi} \int_{\Gamma_I} \frac{\partial Y_j^k(\theta', \phi') r'^j}{\partial n} v_I(Q) \, \mathrm{d}\Gamma \tag{48}$$

In practical computations, the summations in the infinite series (42), (43), (46) and (47) are truncated after p terms. The estimation of errors involved in truncated summations as well as their proofs can be found in Reference [14].

We have described the process of summations (34) and (35), for a cluster of nodes contained in a leaf B_{far}, with respect to a node $\mathbf{s}_I$ included in another leaf B_{local} which is well separated from B_{far}. In the fast multipole algorithm, this summation is done in a more efficient way rather than performed separately for each pair of well separated leaves. We sum instead the multipole moments about a box centre from all of its children before converting them into local moments, and sum all the local moments before shifting them to the centres of its children. More specifically, the multipole and local moments are orchestrated in the tree-structured hierarchy of boxes. Multipole moments are either combined to form multipole moments which represent distributions in a greater portion of the domain, or transformed into local moments. Multipole moments for the boxes are accumulated from leaves of the tree to the root (*upward pass*); and local moments are distributed from the root to the leaves for evaluation of far field influence at the nodes in the leaf (*downward pass*). This is accomplished in order of N operations. The algorithm is summarized as follows.

FM–HBNM ALGORITHM

Initialization

Step 1: Choose a smallest box that contains the entire computational domain, and use it as the root box of the hierarchical decomposition of the domain.

Choose the maximum number of nodes contained in a leaf.

Construct the hierarchy of boxes using an oct-tree data structure.

Step 2: Choose the desired multipole expansion order p. With each box that $l > 1$, associate a $p \times p$ matrix which describes the multipole moments M_n^m about the box centre. With each box that $l > 2$, associate a $p \times p$ matrix which describes the local moments L_j^k about the box centre. With each leaf associate two matrices of size $p \times p \times N_b$, which describe S_j^k and $\partial S_j^k / \partial n$, where N_b denotes the number of nodes contained in that leaf. Associate other two matrices U_L and V_L of size $N_b \times N_n$ to the leaf, which describe the near-field coefficients of U and V, with N_n being the number of nodes contained in the neighbourhood of that leaf.

Step 3: For each leaf, compute U_L, V_L, S_j^k and $\partial S_j^k / \partial n$ by Equations (23), (24), (44) and (48), respectively.

Upward pass

Step 4: For an iteration vector $\mathbf{x}'$, form multipole moments M_n^m about the centre of each leaf from all the nodes included in that leaf by means of Equation (40).

Step 5: For levels l from the finest level to level 2;

Form multipole moments M_n^m about the centre of each box at level l by merging multipole moments from its children using Equation (45) (M2M in Figure 4).

Downward pass

Step 6: For levels l from level 2 to the finest level,

For each box i at level l,

Convert the multipole moments M_n^m of each box j in the *interaction list* of box i to a local expansion about the centre of box i, using Equation (43) (M2L in Figure 4).

If $l \geqslant 3$, then shift the local expansion of i's parent to itself, using Equation (46) (L2L in Figure 4).

Add these two local expansions together.

Step 7: For each node in each leaf,

Compute the far field influence using Equations (42) and (47).

Compute the near field influence deduced by the nodes contained in the neighbourhood of the leaf directly.

Add the two parts.

In the above algorithm, the multipole and local moments associated with a box at each level are reused to the full extent. For example, in computation of Equations (42) and (47), the local expansion of a leaf is reused for all the nodes belonging to the leaf; and in the conversion of multipole moments into a local expansion (Equation (43)), the multipole moments of a box is repeatedly used for the boxes whose interaction list contains that box, and so on. This is why FMM can speed up the matrix–vector multiplication. An estimation of computational cost of the algorithm can be found in References [13, 14].

4. BINARY TREE DATA STRUCTURE FOR MLS APPROXIMATION

In the previous section, we have discussed how to reduce the computational cost involved in the solution of the system of equations using GMRES and FMM. We can see that the FMM

Table I. Flowchart of original MLS approximation.

1. Choose a finite number of nodes on the panel
2. Determine the support sizes, $\hat{d}_I$ and $\hat{d}'_I$, of the weight function for each node
3. Loop over all nodes located on the panel
   a. determine the nodes $\mathbf{s}_I$ that $w_I(\mathbf{s})>0$
   b. if $w_I(\mathbf{s})>0$, calculate the right hand side of Equation (10)
   c. add contributions to $\mathbf{A}(\mathbf{s})$
4. Solve the inversion of $\mathbf{A}(\mathbf{s})$
5. Loop over all nodes located on the panel
   For each node $\mathbf{s}_I$ that $w_I(\mathbf{s})>0$, calculate $w_I(\mathbf{s})\mathbf{p}(\mathbf{s}_I)$ and then $\Phi_I(\mathbf{s})$ using Equations (11) and (9)

concerns matrices $\mathbf{U}$ and $\mathbf{V}$, only, while leaves matrix $\mathbf{H}$ intact. The matrix $\mathbf{H}$ involves the MLS shape functions. There are two usages of $\mathbf{H}$ in Hybrid BNM. One is for computing the right hand side vector of Equations (21) and (22), while the other for solving the boundary unknowns $\hat{\mathbf{u}}$ and $\hat{\mathbf{q}}$ by Equations (32) and (33) after $\mathbf{x}$ has been solved. Since the MLS approximation in Hybrid BNM is conducted separately on individual panels, the matrix $\mathbf{H}$, unlike $\mathbf{U}$ and $\mathbf{V}$, is diagonally blocked. Even so, for a panel with a large number of nodes located, the size of the corresponding block will be extremely large, and the evaluation of the shape functions in Equation (25) can be expensive. In order to solve this problem, we use a binary tree data structure to speed up MLS approximation and reduce the memory requirements for storing matrix $\mathbf{H}$.

In BEM, the number of shape functions for an evaluation point equals to the number of nodes of an element, while in Hybrid BNM, it equals to the total number of nodes on the panel. Although most of them are zero due to the compact support of the weight function of each node, we do not know which of them is zero before it is computed. The reason is, in the input data structure of Hybrid BNM, there is no information of connectivity between the nodes. According to Equations (9)–(12), the flowchart of the original MLS approximation at a panel is shown in Table I.

Note that in Table I, computation of the shape functions for each evaluation point needs to loop over all the nodes located on the panel to check the condition, $w_I(\mathbf{s})>0$. This check is time consuming especially when the number of nodes is very large. Moreover, the locations of the non-zero entries in every row of matrix $\mathbf{H}$ (see Equation (25)) depend upon the nodes located inside the domain of influence of the source node. If the shape and size of the domain of influence for all of the nodes are taken to be the same as each other, it may be easy to see that the resulting block of $\mathbf{H}$ becomes banded with non-zero entries being symmetrically and sparsely located with unsymmetrical values. However, since we cannot determine the bandwidth in advance, we have to store the entire block in memory. This can lead to exhaustion of computer memory.

In order to overcome the above two shortcomings, we adapt the tree data structure used in FMM and apply it to MLS approximation. Because the MLS approximation in Hybrid BNM for 3D problems is carried out on 2D panels, we use a binary tree data structure to represent a hierarchical partitioning of a panel with cells. Because further that the panel is represented in parametric form, we subdivide the panel in parametric space. We associate a cell with the

Table II. Flowchart of MLS approximation with a binary tree data structure.

1. Choose a finite number of nodes on the panel
2. Determine the support sizes, $\hat{d}_I$ and $\hat{d}'_I$, of the weight function for each node
3. Create the binary tree data structure to subdivide the panel into hierarchical cells in the parametric space
4. Find the leaf L_c that includes the evaluation point $\mathbf{s}$
5. Loop over the nodes included in the neighbourhood of L_c
   a. determine the nodes $\mathbf{s}_I$ that $w_I(\mathbf{s})>0$
   b. if $w_I(\mathbf{s})>0$, calculate the right hand side of Equation (10)
   c. add contributions to $\mathbf{A}(\mathbf{s})$
6. Solve the inversion of $\mathbf{A}(\mathbf{s})$
7. Loop over the nodes included in the neighbourhood of L_c
   For each node $\mathbf{s}_I$ that $w_I(\mathbf{s})>0$, calculate $w_I(\mathbf{s})\mathbf{p}(\mathbf{s}_I)$ and then $\Phi_I(\mathbf{s})$ using Equations (11) and (9)

following parameters:

- $Centre.s_1$ denotes the value of parametric co-ordinate of the centre of the cell in s_1 direction.
- $Centre.s_2$ denotes the value of parametric co-ordinate of the centre of the cell in s_2 direction.
- $H.s_1$ denotes the side length of the cell in s_1 direction.
- $H.s_2$ denotes the side length of the cell in s_2 direction.
- $D\max.s_1$ denotes the maximum value of $\hat{d}_I$ among the nodes included in the cell.
- $D\max.s_2$ denotes the maximum value of $\hat{d}'_I$ among the nodes included in the cell.

Consider the biggest cell, which contains the entire panel and refer this cell as the level 0 or root cell. Given a subdivision S of the computation cell, if $H.s_1$ is bigger than $D\max.s_1$, we subdivide S into two equal cells in s_1 direction. Similarly, we subdivide it into two equal cells in s_2 direction if $H.s_2$ is bigger than $D\max.s_2$. This process is recursively repeated down from the root cell to some finest level. We refer the cells at the finest level as *leaves*. In the next step, we create a *neighbour list* for every leaf. For a leaf L, we loop over all other leaves L_i. If the distances between L_i and L in both s_1 and s_2 directions are smaller than $D\max.s_1$ and $D\max.s_2$ associated with L_i, L_i is treated as a neighbour of L's and added to L's neighbour list. A leaf is also a neighbour of itself. Now, instead of creating an $n \times n$ square block of $\mathbf{H}$, we associate each leaf a $j \times k$ sub-matrix $\mathbf{h}_{jk}$, where n is the total number of nodes on the panel; j denotes the number of nodes included in the leaf; and k the number of nodes included in the neighbours of the leaf. This scheme saves computer memory considerably.

With the binary tree data structure, the routine for computing the shape functions becomes that shown in Table II.

The loop for checking weight functions in step 5a and 7 in Table II contains the nodes that are in the neighbourhood of a leaf, only. When the total number of nodes at the panel is large, the saving in CPU time by the new algorithm will be significant.

5. NUMERICAL RESULTS

The proposed techniques have been implemented in C++. Four numerical examples are presented to demonstrate the performance of the method. The first three examples consider relatively

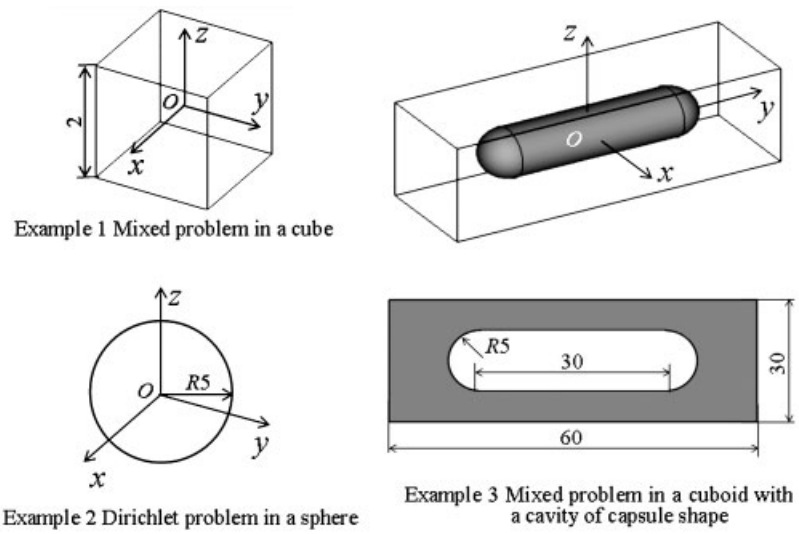

Figure 5. Geometries and dimensions for the first three examples.

simple geometries, namely, a cube, a sphere and a cuboid including a void of capsule shape. Dimensions of the three geometries are illustrated in Figure 5. In order to assess the accuracy of the proposed approach the following field distribution is used as the exact solution:

$$u = x^3 + y^3 + z^3 - 3yx^2 - 3xz^2 - 3zy^2 \tag{49}$$

The error is estimated on sample points using the formula from Reference [4]

$$e = \frac{1}{|u|_{\max}} \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} \left(u_i^{(e)} - u_i^{(n)}\right)^2} \tag{50}$$

where N_s is the number of sample points; $|u|_{\max}$ is the maximum value of u over all sample points; the superscripts (e) and (n) refer to the exact and numerical solutions, respectively.

In the cube model, potential boundary conditions are imposed on the top and bottom faces and normal flux boundary on other faces according Equation (49). The relative error is evaluated over 21 sample points uniformly distributed along a line segment from $(0, 0, 0)$ to $(1, 1, 1)$ using Equation (50). In the sphere the model, potential boundary conditions are imposed at the entire surface. The relative error is evaluated over 21 sample points uniformly distributed on a diameter from $(-5, 0, 0)$ to $(5, 0, 0)$. For the cuboid, potential boundary conditions are imposed on the two end faces and normal flux boundary conditions on the other faces, including the side faces and the inner surface of the cavity. The relative error is evaluated over 42 sample points uniformly distributed on a line all through the cuboid (from $(7, -30, 0)$ to $(7, 30, 0)$).

Computations are performed on a desktop computer with an Intel(R) Pentium(R) 4 CPU (1.99 GHz). The restarted preconditioned GMRES is employed with the preconditioner being the inverse of the blocked diagonal matrix corresponding to the nodes in leaves, according to Reference [16]. Following Reference [17], we truncate all the infinite expansions after

Table III. Timing results for FM–HBNM with nodes distributed on the surface of a cube.

N	Levels	Iterations	T_{FMM}	T_{dir}(s)	E_{FMM}	E_{dir}
600	3	13	63	2	5.5×10^{-3}	5.4×10^{-3}
2400	4	16	78	88	2.7×10^{-3}	2.7×10^{-3}
5400	4	18	446	1019	1.9×10^{-3}	1.8×10^{-3}
15 000	5	20	706	(21 841)	1.2×10^{-3}	
60 000	6	27	2782	(1 397 805)	5.9×10^{-4}	
101 400	6	32	11 988	(6 746 939)	4.5×10^{-4}	
153 600	6	35	13 471	(23 451 280)	4.0×10^{-4}	

Table IV. Timing results for FM–HBNM with nodes distributed on the surface of a sphere.

N	Levels	Iterations	T_{FMM}	T_{dir}(s)	E_{FMM}	E_{dir}
540	3	16	54	1	3.8×10^{-3}	3.8×10^{-3}
2088	3	17	83	47	1.2×10^{-3}	1.2×10^{-3}
4664	4	20	458	515	9.1×10^{-4}	9.1×10^{-4}
8252	4	21	483	(2852)	7.3×10^{-4}	
18 496	5	27	2267	(32 119)	5.0×10^{-4}	
32 824	5	27	2463	(179 517)	3.8×10^{-4}	
73 664	6	39	10 221	(2 029 072)	2.6×10^{-4}	

$p = 10$, set the maximum number of boundary nodes in a leaf box to be 60, and terminate the iteration when the relative error is less than 10^{-6}. For comparison purposes, the models have also been solved by means of the original Hybrid BNM with a direct solver (Gaussian elimination method) in the cases where it is capable of solving it. For the three geometries, computations have been carried out with different numbers of nodes uniformly scattered on the bounding surface of the bodies. The results of our experiments are summarized in Tables III–V. In each table, the first, second and third column list the number of nodes, number of levels used in the multipole hierarchy, and number of iterations of GMRES, respectively. Columns four and five indicate the times required for FM–HBNM and the original Hybrid BNM, respectively. As the direct calculations are restricted to about 5400 nodes due to the limitation of the hardware, the stated times in parentheses are estimated by extrapolation. In the sixth and seventh column, the relative errors of FM–HBNM and the original Hybrid BNM are presented. The results demonstrate that FM–HBNM is extremely effective for large-scale computation. The computational time of FM–HBNM is nearly proportional to the problem size. From Table III, it is seen that FM–HBNM has an advantage over the original Hybrid BNM when the number of unknowns is bigger than around 2400.

In order to further show the advantages of FM–HBNM, the fourth example deals with a complicated geometry, where a cuboid including many cavities in the shape of sinusoidal tubes. The geometry and its main dimensions are shown in Figure 6. The radii of all the tubes are identical and equal to 0.5. A Dirichlet problem is solved for which the essential boundary conditions are imposed on all the surfaces (including the cavities) according to Equation (49).

Table V. Timing results for FM–HBNM with nodes distributed on the surface of a cuboid.

N	Levels	Iterations	T_{FMM}	T_{dir}(s)	E_{FMM}	E_{dir}
937	4	14	31	5	2.6×10^{-3}	2.6×10^{-3}
1400	4	14	179	15	2.0×10^{-3}	2.0×10^{-3}
4598	4	14	182	523	4.6×10^{-3}	4.6×10^{-4}
14 091	5	17	773	(15 053)	3.5×10^{-4}	
56 186	6	21	3358	(954 287)	2.1×10^{-4}	
106 960	7	25	5692	(6583 532)	1.7×10^{-4}	
155 810	7	26	12 396	(20 350 766)	1.2×10^{-4}	
375 708	8	35	21 388		6.3×10^{-5}	
620 708	8	39	53 840		5.7×10^{-5}	

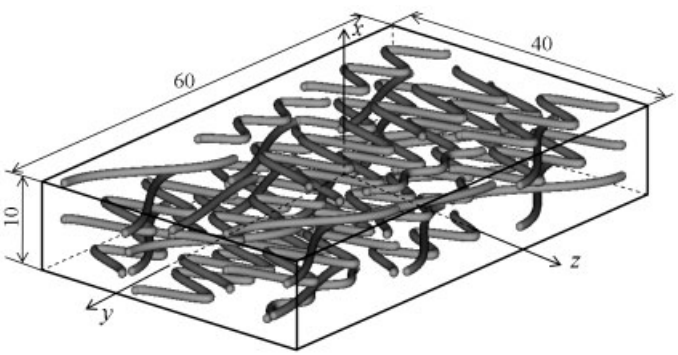

Figure 6. Geometry and dimensions for the fourth example.

Table VI. CPU Time and relative error of nodal values for the fourth example.

Number of nodes	108 960	137 370	159 680	192 500	255 180
Relative error of nodal values	6.42%	3.81%	3.35%	2.33%	1.91%
CPU time (s)	25 386	37 840	43 058	55 670	67 725

Computations are performed for five nodal arrangements, namely, 108 960 nodes, 137 370 nodes, 159 680 nodes, 192 500 nodes and 255 180 nodes, which are uniformly distributed on all the surfaces. The relative errors of nodal values of normal flux, evaluated using Equation (50) with u being replaced by q and N_s being the total number of nodes, are presented in the second row of Table VI. The third row of Table VI lists the computational time used for solving the problem. Figure 7 shows the numerical results, obtained with the fifth nodal arrangement, of u and its y-derivative evaluated at 31 sample points uniformly spaced on the line segment from $(1, -30, 0.8)$ to $(1, 30, 0.8)$. Results in both Table VI and Figure 7 demonstrate that high accuracy and efficiency can be achieved by the FM–HBNM. From Figure 7 it is seen that

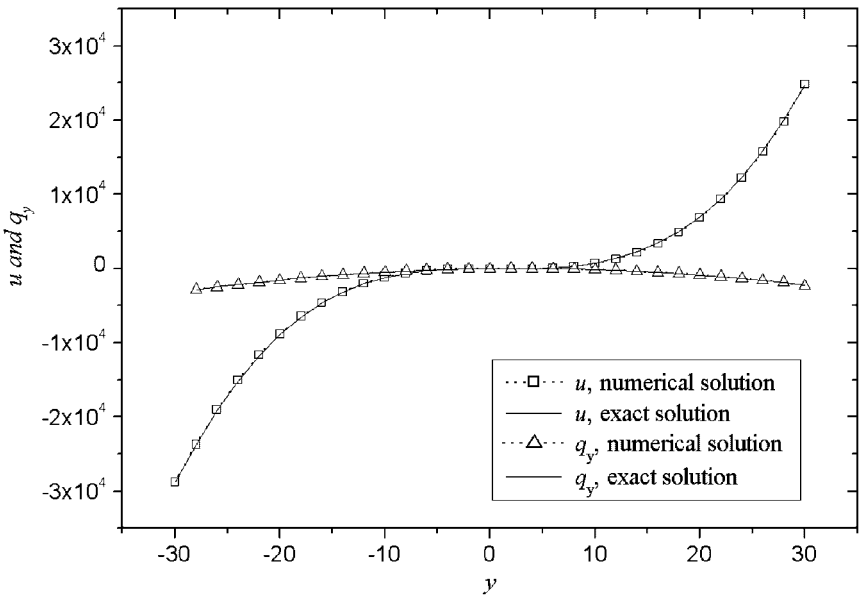

Figure 7. Results of potential and its gradient q_y ($\equiv \partial u/\partial y$) along an internal segment (from $(1, -30, 0.8)$ to $(1, 30, 0.8)$).

no 'boundary layer effect' appears. We should point out here that it is almost impossible to obtain a reasonable discretization with finite elements for this geometry. In an implementation of BEM, much effort is required to discretize all the surfaces with high quality boundary elements. However, in our method, we just need to define the surfaces in their parametric representations, which can be done very easily.

6. CONCLUSIONS AND DISCUSSION

The FMM has been incorporated into the Hybrid BNM. The combined approach has been successfully applied for solving problems in 3D potential theory. Numerical examples demonstrate the nearly linear complexity of the FM–HBNM. High accuracy has been achieved.

The FM–HBNM retains the advantages of both the meshless method and the fast solver. It not only results in considerable savings in computing time and memory, but also substantially simplifies the discretization tasks for problems with complicated geometries. Therefore, the proposed method is especially applicable for large-scale simulations of bodies with intrigue geometries, such as carbon nanotube based composites [21, 22].

It is worth noting that, in the present paper, the Hybrid BNM is combined with the original FMM [12]. Actually, Greengard and Rokhlin have developed a new version of FMM [13], in which they use a new diagonal form for translation operators and further enhance the efficiency of FMM. To incorporate the new FMM into the Hybrid BNM is a subject for future research.

ACKNOWLEDGEMENTS

The support of the CLUSTER of Ministry of Education, Culture, Sports, Science and Technology (Japan) is gratefully acknowledged.

REFERENCES

1. Belytschko T, Lu YY, Gu L. Element free Galerkin methods. *International Journal for Numerical Methods in Engineering* 1994; **37**:229–256.
2. Belytschko T, Krongauz Y, Organ D, Fleming M, Krysl P. Meshless methods: an overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:3–47.
3. Atluri SN, Zhu T. A new meshless local Petrov–Galerkin approach in computational mechanics. *Computational Mechanics* 1998; **22**:117–127.
4. Mukherjee YX, Mukherjee S. The boundary node method for potential problems. *International Journal for Numerical Methods in Engineering* 1997; **40**:797–815.
5. Zhang JM, Yao ZH, Li H. A hybrid boundary node method. *International Journal for Numerical Methods in Engineering* 2002; **53**:751–763.
6. Zhang JM, Tanaka M, Matsumoto T. Meshless analysis of potential problems in three dimensions with the hybrid boundary node method. *International Journal for Numerical Methods in Engineering* 2004; **59**:1147–1160.
7. Zhang JM, Yao ZH. Meshless regular hybrid boundary node method. *Computer Modeling in Engineering and Sciences* 2001; **2**:307–318.
8. Zhang JM, Yao ZH, Tanaka M. The meshless regular hybrid boundary node method for 2-D linear elasticity. *Engineering Analysis with Boundary Elements* 2003; **27**:259–268.
9. Zhang JM, Yao ZH. The regular hybrid boundary node method for three-dimensional linear elasticity. *Engineering Analysis with Boundary Elements* 2004; **28**:525–534.
10. Zhang JM, Yao ZH. Analysis of 2-D thin structures by the meshless regular hybrid boundary node method. *Acta Mechanica Sinica* 2002; **15**:36–44.
11. Rokhlin V. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics* 1985; **60**:187–207.
12. Greengard L, Rokhlin V. A fast algorithm for particles simulations. *Journal of Computational Physics* 1987; **73**:325–348.
13. Greengard L, Rokhlin V. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica* 1997; **6**:229–269.
14. Greengard L. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press: Cambridge, 1988.
15. Nabors K, Korsmeyer FT, Leighton FT, White J. Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory. *SIAM Journal on Scientific Computing* 1994; **15**:713–735.
16. Nishida T, Hayami K. Application of the fast multipole method to the 3D BEM analysis of electron guns. In *Boundary Elements XIX*, Marchettia M, Brebbia CA, Aliabadi MH (eds). Computational Mechanics Publications: Southampton, 1997; 613–622.
17. Yoshida K, Nishimura N, Kobayashi S. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering* 2001; **50**:525–547.
18. DeFigueredo TGB, Brebbia CA. A new hybrid displacement variational formulation of BEM for elastostatics. In *Advances in Boundary Elements*, Brebbia CA, Conner JJ (eds), vol. 1. Computational Mechanics Publications: Southampton, 1989; 47–57.
19. Atluri SN, Kim HG, Cho JY. A critical assessment of the truly meshless local Petrov–Galerkin (MLPG), and local boundary integral equation (LBIE) methods. *Computational Mechanics* 1999; **24**:348–372.
20. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear system. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**:856–869.
21. Zhang JM, Tanaka M, Matsumoto T, Guzik A. Heat conduction analysis in bodies containing thin-walled structures by means of Hybrid BNM with an application to CNT-based composites. *JSME International Journal*, *Solid Mechanics and Material Engineering* (*Series A*) 2004; **47**:181–188.
22. Zhang JM, Tanaka M, Matsumoto T. A simplified approach for heat conduction analysis of CNT-based nano-composites. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:5597–5609.

