Transactions of JASCOME, Vol. 5 (June 2005), Paper No. 05-062401 **JASCOME**

ACCELERATING MULTI-DOMAIN HYBRID BOUNDARY NODE METHOD WITH FAST MULTIPOLE METHOD

Jianming ZHANG [1)], Masataka TANAKA [2)], Morinobu ENDO [3)]

1) Faculty of Engineering, Shinshu University, (Nagano 380-8553, e-mail: zhangjm@homer.shinshu-u.ac.jp)

2) Faculty of Engineering, Shinshu University, (Nagano 380-8553, e-mail: dtanaka@gipwc.shinshu-u.ac.jp)

3) Faculty of Engineering, Shinshu University, (Nagano 380-8553, e-mail: endo@endomoribu.shinshu-u.ac.jp)

This work presents a fast implementation of the multi-domain hybrid boundary node method (HdBNM) for numerical solution of Laplace's equation. The preconditioned GMRES is employed to solve the overall system of equations. At each iteration step of the GMRES, the matrix-vector multiplication is split into smaller scale ones at the subdomain level, and accelerated by the fast multipole method independently within individual subdomains. The computed matrix-vector products at the subdomain level are then assembled into an overall vector using the equilibrium and continuity conditions at the interfaces. Our method is tested by two benchmark examples for three-dimensional potential problems, and high accuracy and efficiency are observed.

Keywords: meshless method; hybrid boundary node method; multi-domain formulation; fast multipole method

1. Introduction

Meshless techniques to obtain numerical solutions for PDEs without resorting to an element frame have been popular throughout the computational mechanics community for the past two decades. This is because, with mesh-based techniques such as the finite element method (FEM) or the boundary element method (BEM), the task of mesh generation for complex geometries is often time-consuming and prone to errors, and the difficulties with re-meshing in problems involving moving boundaries, large deformations or crack propagation are crucial. Many meshless methods have been proposed so far. Some of the methods are the element free Galerkin method (EFG) [1], the meshless local Petrov-Galerkin (MLPG) approach [2], the boundary node method (BNM) [3] and the hybrid boundary node method (HdBNM) [4-6]. Among these methods, the HdBNM is a truly meshless boundary-only method, which combines the MLS approximation scheme with the hybrid displacement variational formulation. It not only has the advantage of reducing the spatial dimensions by one as BEM, but also does not require any cells either for interpolation of the solution variables or for the boundary integration. In fact, the HdBNM 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, the system matrix of the HdBNM is dense and unsymmetrical. The computational time and memory requirement for directly factoring such system increase respectively with $O(N^3)$ and $O(N^2)$, where N is the total number of degrees of freedom. In order to obtain an efficient algorithm not only in terms of human-labor costs (where mesh generation is avoided) but also in terms of computer costs, we have recently combined the HdBNM with the fast multipole method (FMM) [7-10]. The combined approach (here called FM-HdBNM) reduces both the memory requirement and the total execution count to $O(N)$. Therefore, it is promising for large scale computations. In this paper, we further implement the FMM techniques in a multi-domain formulation of the HdBNM.

Multi-domain formulations are employed when the entire domain under consideration is governed by individual differential equations in different parts and/or constructed of different materials. Besides, in the case of a domain with complicated boundary profile or parallel computation, the domain may be decomposed for better computational efficiency. In a multi-domain solver, the original domain is divided into a finite number of sub-domains, and in each of them the full integral representation formula is applied. At the common interfaces between the adjacent subdomains, the corresponding full matching conditions are enforced. How to satisfy the continuity and equilibrium conditions at the interfaces is one of the important aspects of implementation for a multi-domain algorithm. There are mainly two methods in the literature: the standard multi-domain method [11] and the domain decomposition method [12]. In the standard multi-domain method, the discretized equations corresponding to the subdomains are assembled into a system of equations according the boundary and interface conditions. While the matrices that arise in the single domain formulation are fully populated, the multi-domain formulation leads to overall matrix equations with a sparse blocked structure. In the domain decomposition method, the interface conditions are assumed and then the subdomain problems are solved independently. The modification of the interface condition is usually iterative

using different methodologies, as the Schwarz Neumann-Neumann and Schwarz Dirichlet-Neumann methods. Repetition of the iteration process is continued until convergence. The domain decomposition method allows different type of discretization methods (e.g. BEM and FEM) to be used for a numerical solution of the individual subdomains and coupling between them without accessing to the source codes of the methods. However, it has some relevant parameters to be chosen and the optimal values for these parameters are usually problem-dependent. This arbitrariness represents a disadvantage of the method. In the present paper, we adopt the standard multi-domain method, and make full use of the resultant sparsity of the matrix equations during the solution process. As the sparse structure of the matrix is directly related to the ordering of blocks occurring in the matrix, we use the ordering strategy suggested by Kane [11] to obtain an optimal blocks structure. The preconditioned restarted GMRES is employed to solve the system equations. At each step of the iterations of GMRES, the matrix-vector multiplication is accelerated by the FMM at the subdomain level. Therefore, the FM-HdBNM code for single domain problem can be used directly. The algorithm is implemented through a code written in C++. In the code, an interface class is devised to deal with the equilibrium and continuity conditions at the interfaces. Two benchmark examples of three-dimensional potential problems are investigated. Numerical results demonstrate the accuracy and efficiency of the proposed approach.

2. Multi-domain formulation of HdBNM

In this section, we will derive a multi-domain formulation for solving 3D potential problems. The formulation is obtained by assembling the equations for each single domain into an overall system of equations using the continuity and equilibrium relations along the interfaces between the subdomains. The HdBNM formulation for solving single domain problems has been given in reference [5]. For the sake of simplicity and to allow for a clear presentation of the multi-domain formulation, we consider here three subdomains.

The hybrid boundary node method is based on a modified variational principle, in which there are three independent variables, namely:

- temperature within the domain, ϕ ;
- boundary temperature, $\tilde{\phi}$;
- boundary normal heat flux, $\tilde{q}$.

Suppose further that N nodes are randomly distributed on the bounding surface of subdomain-1. The temperature within the domain is approximated using fundamental solutions as follows:

$$\phi = \sum_{I=1}^{N} \phi_I^s x_I \tag{1}$$

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

$$q = -\kappa_1 \sum_{I=1}^{N} \frac{\partial \phi_I^s}{\partial n} x_I \tag{2}$$

where ϕ_I^s is the fundamental solution with the source at a node $\mathbf{s}_I$; κ_1 is the heat conductivity and x_I are unknown parameters. For 3-D potential problems, the fundamental solution can be written as

$$\phi_I^s = \frac{1}{\kappa_1} \frac{1}{4\pi r(Q, \mathbf{s}_I)} \tag{3}$$

where Q is a field point; $r(Q, \mathbf{s}_I)$ is the distance between Q and $\mathbf{s}_I$.

The boundary temperature and normal heat flux are interpolated by moving least square (MLS) approximation [5]:

$$\tilde{\phi}(\mathbf{s}) = \sum_{I=1}^{N} \Phi_I(\mathbf{s}) \hat{\phi}_I \tag{4}$$

$$\tilde{q}(\mathbf{s}) = \sum_{I=1}^{N} \Phi_I(\mathbf{s}) \hat{q}_I \tag{5}$$

In the foregoing equations, $\Phi_I(\mathbf{s})$ is the shape function of MLS approximation; $\hat{\phi}_I$ and $\hat{q}_I$ are nodal values of temperature and normal flux, respectively.

Using the modified functional variational principle in all local-regions around the boundary nodes, the following set of HdBNM equations can be written for subdomain-1:

$$\mathbf{U}\mathbf{x} = \mathbf{H}\hat{\boldsymbol{\phi}} \tag{6}$$

$$\mathbf{Q}\mathbf{x} = \mathbf{H}\hat{\mathbf{q}} \tag{7}$$

In the above equations, the elements of matrices **U**, **Q** and **H** are given by

$$U_{JI} = \int_{\Gamma_{s_J}} \phi_I^s(Q, \mathbf{s}_I) v_J(Q) d\Gamma \tag{8}$$

$$Q_{JI} = \int_{\Gamma_{s_J}} \frac{\partial \phi_I^s(Q, \mathbf{s}_I)}{\partial n(Q)} v_J(Q) d\Gamma \tag{9}$$

$$H_{JI} = \int_{\Gamma_{s_J}} \Phi_I(Q) v_J(Q) d\Gamma \tag{10}$$

where v_J is a weight function and $\mathbf{s}_I$ is a boundary point, Γ_{s_J} is a regularly shaped local region around a given node $\mathbf{s}_J$ in the parametric representation space of the boundary surface. (For full details of HdBNM refer to [5]).

To assemble equations (5) and (6) into an overall system of equation for the entire domain later, we sort the boundary nodes into three groups: group 1 containing nodes that belong exclusively in subdomain-1, group 2 containing nodes that are on the interface with subdomain-2, and group 3 containing nodes on the interface with subdomain-3. Correspondingly, equations (5) and (6) are partitioned into blocked matrix equations as

$$\begin{bmatrix} \boldsymbol{U}_{11}^1 & \boldsymbol{U}_{12}^1 & \boldsymbol{U}_{13}^1 \\ \boldsymbol{U}_{21}^1 & \boldsymbol{U}_{22}^1 & \boldsymbol{U}_{23}^1 \\ \boldsymbol{U}_{31}^1 & \boldsymbol{U}_{32}^1 & \boldsymbol{U}_{33}^1 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^1 \\ \boldsymbol{x}_2^1 \\ \boldsymbol{x}_3^1 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{H}_1^1 \hat{\boldsymbol{\phi}}_1^1 \\ \boldsymbol{H}_2^1 \hat{\boldsymbol{\phi}}_2^1 \\ \boldsymbol{H}_3^1 \hat{\boldsymbol{\phi}}_3^1 \end{Bmatrix} \tag{11}$$

$$\begin{bmatrix} \boldsymbol{Q}_{11}^1 & \boldsymbol{Q}_{12}^1 & \boldsymbol{Q}_{13}^1 \\ \boldsymbol{Q}_{21}^1 & \boldsymbol{Q}_{22}^1 & \boldsymbol{Q}_{23}^1 \\ \boldsymbol{Q}_{31}^1 & \boldsymbol{Q}_{32}^1 & \boldsymbol{Q}_{33}^1 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^1 \\ \boldsymbol{x}_2^1 \\ \boldsymbol{x}_3^1 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{H}_1^1 \hat{\boldsymbol{q}}_1^1 \\ \boldsymbol{H}_2^1 \hat{\boldsymbol{q}}_2^1 \\ \boldsymbol{H}_3^1 \hat{\boldsymbol{q}}_3^1 \end{Bmatrix} \tag{12}$$

where superscript 1 stands for the subdomain-1; the

subscripts 1, 2, 3 denote that the prescribed quantities are associated with the nodes in groups 1, 2, 3, respectively. The double subscript *ij*, *i*, *j*=1, 2, 3, is used to convey that the pair of nodes $\mathbf{s}_I$ and $\mathbf{s}_J$ in equations (8) and (9), by which the prescribed coefficient matrix blocks are computed, belong to group *i* and *j*, respectively.

Similarly, for subdomain-2 we have

$$\begin{bmatrix} \boldsymbol{U}_{11}^2 & \boldsymbol{U}_{12}^2 & \boldsymbol{U}_{13}^2 \\ \boldsymbol{U}_{21}^2 & \boldsymbol{U}_{22}^2 & \boldsymbol{U}_{23}^2 \\ \boldsymbol{U}_{31}^2 & \boldsymbol{U}_{32}^2 & \boldsymbol{U}_{33}^2 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^2 \\ \boldsymbol{x}_2^2 \\ \boldsymbol{x}_3^2 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{H}_1^2 \hat{\boldsymbol{\phi}}_1^2 \\ \boldsymbol{H}_2^2 \hat{\boldsymbol{\phi}}_2^2 \\ \boldsymbol{H}_3^2 \hat{\boldsymbol{\phi}}_3^2 \end{Bmatrix} \tag{13}$$

$$\begin{bmatrix} \boldsymbol{Q}_{11}^2 & \boldsymbol{Q}_{12}^2 & \boldsymbol{Q}_{13}^2 \\ \boldsymbol{Q}_{21}^2 & \boldsymbol{Q}_{22}^2 & \boldsymbol{Q}_{23}^2 \\ \boldsymbol{Q}_{31}^2 & \boldsymbol{Q}_{32}^2 & \boldsymbol{Q}_{33}^2 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^2 \\ \boldsymbol{x}_2^2 \\ \boldsymbol{x}_3^2 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{H}_1^2 \hat{\boldsymbol{q}}_1^2 \\ \boldsymbol{H}_2^2 \hat{\boldsymbol{q}}_2^2 \\ \boldsymbol{H}_3^2 \hat{\boldsymbol{q}}_3^2 \end{Bmatrix} \tag{14}$$

and for subdomain-3,

$$\begin{bmatrix} \boldsymbol{U}_{11}^3 & \boldsymbol{U}_{12}^3 & \boldsymbol{U}_{13}^3 \\ \boldsymbol{U}_{21}^3 & \boldsymbol{U}_{22}^3 & \boldsymbol{U}_{23}^3 \\ \boldsymbol{U}_{31}^3 & \boldsymbol{U}_{32}^3 & \boldsymbol{U}_{33}^3 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^3 \\ \boldsymbol{x}_2^3 \\ \boldsymbol{x}_3^3 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{H}_1^3 \hat{\boldsymbol{\phi}}_1^3 \\ \boldsymbol{H}_2^3 \hat{\boldsymbol{\phi}}_2^3 \\ \boldsymbol{H}_3^3 \hat{\boldsymbol{\phi}}_3^3 \end{Bmatrix} \tag{15}$$

$$\begin{bmatrix} \boldsymbol{Q}_{11}^3 & \boldsymbol{Q}_{12}^3 & \boldsymbol{Q}_{13}^3 \\ \boldsymbol{Q}_{21}^3 & \boldsymbol{Q}_{22}^3 & \boldsymbol{Q}_{23}^3 \\ \boldsymbol{Q}_{31}^3 & \boldsymbol{Q}_{32}^3 & \boldsymbol{Q}_{33}^3 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^3 \\ \boldsymbol{x}_2^3 \\ \boldsymbol{x}_3^3 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{H}_1^3 \hat{\boldsymbol{q}}_1^3 \\ \boldsymbol{H}_2^3 \hat{\boldsymbol{q}}_2^3 \\ \boldsymbol{H}_3^3 \hat{\boldsymbol{q}}_3^3 \end{Bmatrix} \tag{16}$$

At the interface between subdomain-*i* and *j*, both the temperature and heat flux must be continuous, i.e.,

$$\left\{\boldsymbol{\phi}_i^j\right\} = \left\{\boldsymbol{\phi}_j^i\right\} \tag{17}$$

$$\left\{\boldsymbol{q}_i^j\right\} = -\left\{\boldsymbol{q}_j^i\right\} \tag{18}$$

If we use the same set of nodes distributed on an interface in the discretization for both domains that share the interface, the following relationship exists:

$$\left\{\boldsymbol{H}_i^j\right\} = \left\{\boldsymbol{H}_j^i\right\} \tag{19}$$

Using the continuity conditions, equations (11)-(16) can be assembled into an overall matrix equation:

$$\begin{bmatrix} \boldsymbol{A}_{11}^1 & \boldsymbol{A}_{12}^1 & \boldsymbol{A}_{13}^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{U}_{21}^1 & \boldsymbol{U}_{22}^1 & \boldsymbol{U}_{23}^1 & -\boldsymbol{U}_{11}^2 & -\boldsymbol{U}_{12}^2 & -\boldsymbol{U}_{13}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{U}_{31}^1 & \boldsymbol{U}_{32}^1 & \boldsymbol{U}_{33}^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{U}_{11}^3 & -\boldsymbol{U}_{12}^3 & -\boldsymbol{U}_{13}^3 \\ \boldsymbol{Q}_{21}^1 & \boldsymbol{Q}_{22}^1 & \boldsymbol{Q}_{23}^1 & \boldsymbol{Q}_{11}^2 & \boldsymbol{Q}_{12}^2 & \boldsymbol{Q}_{13}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{21}^2 & \boldsymbol{A}_{22}^2 & \boldsymbol{A}_{23}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{U}_{31}^2 & \boldsymbol{U}_{32}^2 & \boldsymbol{U}_{33}^2 & -\boldsymbol{U}_{21}^3 & -\boldsymbol{U}_{22}^3 & -\boldsymbol{U}_{23}^3 \\ \boldsymbol{Q}_{31}^1 & \boldsymbol{Q}_{32}^1 & \boldsymbol{Q}_{33}^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{11}^3 & \boldsymbol{Q}_{12}^3 & \boldsymbol{Q}_{13}^3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{31}^2 & \boldsymbol{Q}_{32}^2 & \boldsymbol{Q}_{33}^2 & \boldsymbol{Q}_{21}^3 & \boldsymbol{Q}_{22}^3 & \boldsymbol{Q}_{23}^3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{31}^3 & \boldsymbol{A}_{32}^3 & \boldsymbol{A}_{33}^3 \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1^1 \\ \boldsymbol{x}_2^1 \\ \boldsymbol{x}_3^1 \\ \boldsymbol{x}_1^2 \\ \boldsymbol{x}_2^2 \\ \boldsymbol{x}_3^2 \\ \boldsymbol{x}_1^3 \\ \boldsymbol{x}_2^3 \\ \boldsymbol{x}_3^3 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{d}_1^1 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{d}_2^2 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{d}_3^3 \end{Bmatrix} \tag{20}$$

where $\left[\boldsymbol{A}_{ij}^i\right]$ and $\left\{\boldsymbol{d}_i^i\right\}$ are formed by merging $\left[\boldsymbol{U}_{ij}^i\right]$ and $\left[\boldsymbol{Q}_{ij}^i\right]$, $\left\{\boldsymbol{H}_i^i\hat{\boldsymbol{\phi}}_i^i\right\}$ and $\left\{\boldsymbol{H}_i^i\hat{\boldsymbol{q}}_i^i\right\}$, respectively, according to the known boundary conditions. For degrees of freedom with prescribed temperature, the related elements in $\left\{\boldsymbol{H}_i^i\hat{\boldsymbol{\phi}}_i^i\right\}$ are selected into $\left\{\boldsymbol{d}_i^i\right\}$, and the corresponding rows of $\left[\boldsymbol{U}_{ij}^i\right]$ are selected into $\left[\boldsymbol{A}_{ij}^i\right]$; otherwise, elements in $\left\{\boldsymbol{H}_i^i\hat{\boldsymbol{q}}_i^i\right\}$ are selected into $\left\{\boldsymbol{d}_i^i\right\}$, and the corresponding rows in $\left[\boldsymbol{Q}_{ij}^i\right]$ are selected into $\left[\boldsymbol{A}_{ij}^i\right]$.

The set of equation (20) is solved for the unknown parameters $\boldsymbol{x}$, then, by back-substitution into equations (11)-(16), the boundary unknowns are obtained either on the interfaces or the external boundary surfaces.

The blocked matrix in equation (20) is actually hyper-matrix with smaller matrices as entries. The zero blocks in equation (20) give the equation a very beneficial characteristic, i.e. sparsity. To simply send the whole matrix to an equation-solving subroutine would be extremely inefficient. Techniques for banded or variable-banded matrix equation solving are also ineffective because of the lack of symmetry. To capitalize on the special structure of these sparse blocked matrices, we choose an iterative solver, i.e. GMRES, to solve it in this study. However, for the conventional GMRES, both the computational time and memory size required to store the coefficient matrix are proportional to n^2, where n is the total number of unknowns in the overall system of equations. This limits the method to relatively small scale problems. Accelerating the equation solution process with Fast Multipole techniques is necessary for solving large scale problems.

3. Accelerating multi-domain HdBNM with FMM

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 in the methods based on Green's functions or fundamental solutions. The FMM uses multipole expansions (in term of series) to approximate the effects of a distant group of particles (nodes in HdBNM) on a local group, and thus achieves faster summation. 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. We have implemented the FMM techniques in the HdBNM for single domain problems. In this paper, we will focus on how to accelerate the solution of equation (20).

When an iterative solver is employed to solve a linear system, the most time-consuming part of the solution process is the calculation of the matrix-vector product at each iteration step. Taking an iteration vector into account and considering equations (11)-(16), we suppose that

$$\begin{bmatrix} \boldsymbol{U}_{11}^i & \boldsymbol{U}_{12}^i & \boldsymbol{U}_{13}^i \\ \boldsymbol{U}_{21}^i & \boldsymbol{U}_{22}^i & \boldsymbol{U}_{23}^i \\ \boldsymbol{U}_{31}^i & \boldsymbol{U}_{32}^i & \boldsymbol{U}_{33}^i \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1'^i \\ \boldsymbol{x}_2'^i \\ \boldsymbol{x}_3'^i \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\phi}_1^i \\ \boldsymbol{\phi}_2^i \\ \boldsymbol{\phi}_3^i \end{Bmatrix} \tag{21}$$

and

$$\begin{bmatrix} \boldsymbol{Q}_{11}^i & \boldsymbol{Q}_{12}^i & \boldsymbol{Q}_{13}^i \\ \boldsymbol{Q}_{21}^i & \boldsymbol{Q}_{22}^i & \boldsymbol{Q}_{23}^i \\ \boldsymbol{Q}_{31}^i & \boldsymbol{Q}_{32}^i & \boldsymbol{Q}_{33}^i \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1'^i \\ \boldsymbol{x}_2'^i \\ \boldsymbol{x}_3'^i \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{q}_1^i \\ \boldsymbol{q}_2^i \\ \boldsymbol{q}_3^i \end{Bmatrix} \tag{22}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{q}$ are result vectors of the products. Then, the

overall matrix-vector product in equation (20) can be obtained by

$$\begin{bmatrix} \boldsymbol{A}_{11}^{1} & \boldsymbol{A}_{12}^{1} & \boldsymbol{A}_{13}^{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{U}_{21}^{1} & \boldsymbol{U}_{22}^{1} & \boldsymbol{U}_{23}^{1} & -\boldsymbol{U}_{11}^{2} & -\boldsymbol{U}_{12}^{2} & -\boldsymbol{U}_{13}^{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{U}_{31}^{1} & \boldsymbol{U}_{32}^{1} & \boldsymbol{U}_{33}^{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{U}_{11}^{3} & -\boldsymbol{U}_{12}^{3} & -\boldsymbol{U}_{13}^{3} \\ \boldsymbol{Q}_{21}^{1} & \boldsymbol{Q}_{22}^{1} & \boldsymbol{Q}_{23}^{1} & \boldsymbol{Q}_{11}^{2} & \boldsymbol{Q}_{12}^{2} & \boldsymbol{Q}_{13}^{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{21}^{2} & \boldsymbol{A}_{22}^{2} & \boldsymbol{A}_{23}^{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{U}_{31}^{2} & \boldsymbol{U}_{32}^{2} & \boldsymbol{U}_{33}^{2} & -\boldsymbol{U}_{21}^{3} & -\boldsymbol{U}_{22}^{3} & -\boldsymbol{U}_{23}^{3} \\ \boldsymbol{Q}_{31}^{1} & \boldsymbol{Q}_{32}^{1} & \boldsymbol{Q}_{33}^{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{11}^{3} & \boldsymbol{Q}_{12}^{3} & \boldsymbol{Q}_{13}^{3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}_{31}^{2} & \boldsymbol{Q}_{32}^{2} & \boldsymbol{Q}_{33}^{2} & \boldsymbol{Q}_{21}^{3} & \boldsymbol{Q}_{22}^{3} & \boldsymbol{Q}_{23}^{3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{31}^{3} & \boldsymbol{A}_{32}^{3} & \boldsymbol{A}_{33}^{3} \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1'^1 \\ \boldsymbol{x}_2'^1 \\ \boldsymbol{x}_3'^1 \\ \boldsymbol{x}_1'^2 \\ \boldsymbol{x}_2'^2 \\ \boldsymbol{x}_3'^2 \\ \boldsymbol{x}_1'^3 \\ \boldsymbol{x}_2'^3 \\ \boldsymbol{x}_3'^3 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\phi}_1^1 \text{ or } \boldsymbol{q}_1^1 \\ \boldsymbol{\phi}_2^1 - \boldsymbol{\phi}_1^2 \\ \boldsymbol{\phi}_3^1 - \boldsymbol{\phi}_1^3 \\ \boldsymbol{q}_2^1 + \boldsymbol{q}_1^2 \\ \boldsymbol{\phi}_2^2 \text{ or } \boldsymbol{q}_2^2 \\ \boldsymbol{\phi}_3^2 - \boldsymbol{\phi}_2^3 \\ \boldsymbol{q}_3^1 + \boldsymbol{q}_1^3 \\ \boldsymbol{q}_3^2 + \boldsymbol{q}_2^3 \\ \boldsymbol{\phi}_3^3 \text{ or } \boldsymbol{q}_3^3 \end{Bmatrix} \tag{23}$$

The computational costs for the right hand side of equation (23) are trivial, and can be ignored. The summations in equations (21) and (22) can be accelerated by FMM within each single subdomain independently. In the above solution procedure, the coefficient matrix in equation (20) needs never be formed, and its use is purely symbolic. The matrix-vector product in equation (20) is divided into smaller scale ones at the subdomain level, thus making the fullest use of the sparsity pattern of the coefficient matrix, as consideration of the empty blocks is completely avoided.

The accelerated summation process by FMM for the sums in equations (21) and (22) is exactly the same as that in the FM-HdBNM for single domain problems. We create a hierarchical space decomposition tree for each subdomain. All the computer subroutines for single domain problems can be exploited here directly. We have described the algorithm of FM-HdBNM for single domain problems in references [6]. To avoid repetition, we will not discuss it here again. The reader is referred to the paper [6] for further details.

On the implementation of the above algorithm, we remark the following two aspects:

1. The sparsity pattern (population of the blocks) of the coefficient matrix in equation (20) has a severe impact on the condition number of the matrix, and thus on the solution procedure especially when an iterative equation solver is employed. The sparsity pattern of the system equation is determined by the ordering of unknowns. In order that the nonempty blocks in the overall system are as close to the main diagonal as possible, we use the particular ordering suggested by Kane [11]. The order is determined by listing all permutations of two subdomains as shown below:

$$11 \;\; 12 \;\; 13 \;\; 21^{*} \;\; 22 \;\; 23 \;\; 31^{*} \;\; 32^{*} \;\; 33$$

For permutations where the first digit is less than the second digit, blocks of potential are generated; otherwise, blocks of normal flux are generated. The permutations associated with blocks of normal flux are shown with an asterisk in the above list.

2. The selection of a good preconditioner for the GMRES is crucial for its convergence and computing efficiency. It is even more so with the multi-domain formulation, since the population of the overall equation matrix is no longer diagonally dominated. In this study as a primary step, we just simply use a block diagonal preconditioner that is obtained by inversing the diagonal blocked sub-matrices. These sub-matrices are the further smaller diagonal sub-blocks of the main diagonal blocked matrices, namely $\boldsymbol{A}_{11}^{1}$, $\boldsymbol{U}_{22}^{1}$, $\boldsymbol{U}_{33}^{1}$, $\boldsymbol{Q}_{11}^{2}$, $\boldsymbol{A}_{22}^{2}$, $\boldsymbol{U}_{33}^{2}$, $\boldsymbol{Q}_{11}^{3}$, $\boldsymbol{Q}_{22}^{3}$ and $\boldsymbol{A}_{33}^{3}$, in equation (20). The sub-blocks are formed according to the leaves of the hierarchical decomposition tree. More precisely, if both the nodes $\mathbf{s}_I$ and $\mathbf{s}_J$ in equation (8) or (9) reside in the same leaf, the entry U_{IJ} or Q_{IJ} is selected into the corresponding sub-block. This preconditioner has been proposed by Nishida and Hayami [9] and adopted by Yoshida and Nishimura [10] for solving single-domain problems with FMM. This preconditioner may not be efficient for multi-domain problems. Developing other forms of preconditioners is an important subject of future research.

4. Test problems

The proposed techniques have been implemented in a code written in C++ and tested with two benchmark problems. All computations are carried out on the same desktop computer with an Intel(R) Pentium(R) 4 CPU (1.99GHz). Concerning the FMM and GMRES, we truncate all the infinite expansions after 10 terms, 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}. To assess the accuracy of the method, we calculate the relative error of nodal values of temperature using the following 'global' L_2 norm:

$$err = \frac{1}{|u|_{\max}} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(u_i^{(e)} - u_i^{(n)} \right)^2} \tag{24}$$

where u_i represents nodal values of temperature ϕ or normal flux q, and $|u|_{\max}$ is the maximum value among the nodal values; n is the total number of nodes; the superscripts (e) and (n) refer to the exact and numerical solutions, respectively.

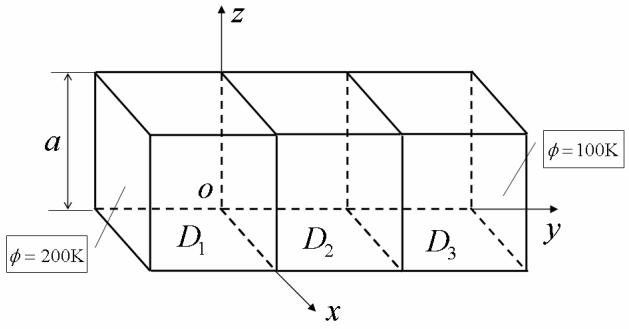


Figure 1. Geometry of the domain consisting of three cubes.

4.1 Three cubes with different material properties

A simple heat conduction problem is first considered. The domain of the problem consists of three equal cubes with different thermal conductivities (Figure 1). The side length of the cubes is $a = 1$ m. The used heat conductivities for cubes D_1, D_2 and D_3 are $\kappa_1 = 1.0$ W/mK, $\kappa_2 = 3.0$ W/mK and $\kappa_3 = 2.0$ W/mK, respectively. A uniform temperature of 200K is imposed at the left end face of cube D_1 and 100K at the right end face of cube D_3. All other outer surfaces are

prescribed as heat flux free. For this problem, the following exact solution is available:

$$\phi = \begin{cases} (1600\text{-}600\times y)/11, & -1 \le y < 0 \\ (1600\text{-}200\times y)/11, & 0 \le y < 1 \\ (1700\text{-}300\times y)/11, & 1 \le y \le 2 \end{cases} \quad (25)$$

We naturally treat each cube as a subdomain and perform computations on five node arrangements, namely, 10×10, 20×20, 40×40, 80×80 and 160×160 nodes uniformly distributed at each square surface of a subdomain. Results are summarized in Tables 1, in which the first and second columns list the number of nodes used on one square surface and the total number of nodes; the third and fourth columns list the number of iterations of GMRES and the total times for solving the system equations. In fifth and sixth columns, the relative errors of nodal values of temperature and normal flux are presented.

Table 1. Results for the domain consisting of three cubes

$k\times k$	*DOFs*	*Its*	T (s)	err_{temp}	err_{flux}
10×10	1800	18	313	1.2×10^{-5}	7.6×10^{-3}
20×20	7200	21	381	5.2×10^{-6}	3.6×10^{-3}
40×40	28800	24	2461	2.7×10^{-6}	1.8×10^{-3}
80×80	115200	32	13578	1.8×10^{-6}	1.3×10^{-3}
160×160	460800	44	72799	8.9×10^{-8}	1.2×10^{-4}

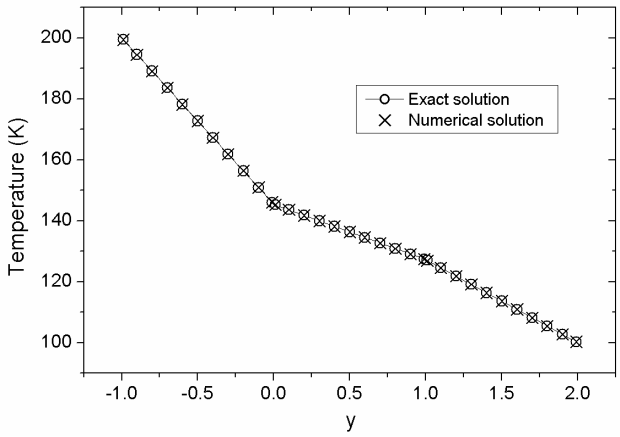


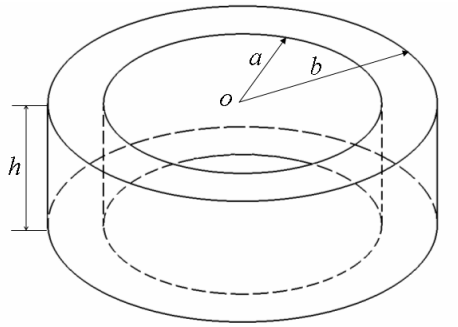
Figure 2. Temperature distribution along the central line.

Numerical results obtained for the temperature with 10×10 nodes on each square face, together with the exact solutions, along the central line from (0.5, -1.0, 0.5) to (0.5, 2.0, 0.5) are presented in Figure 2. The tabulated results show that our algorithm is capable of performing large-scale multi-domain computations. Highly accurate results are obtained with a small number of boundary nodes, and improved with increasing number of nodes used. The high accuracy is also demonstrated in Figure 2, where the numerical results agree excellently with the exact solution.

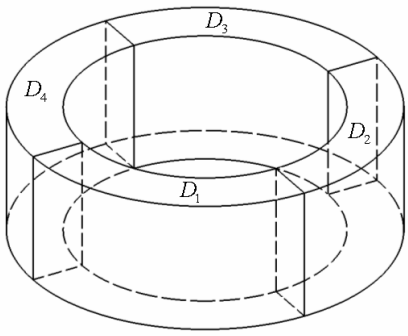
4.2 A cylinder of uniform material

The second example deals with a thick cylinder of uniform material (Figure 3). Dimensions are given as $a=5$, $b=7$ and $h=5$. We use this example to compare the efficiencies between the multi-domain and single domain solution strategies under the circumstance that FMM is employed to accelerate the equation solution. For this purpose, we first model the cylinder as a single domain (Figure 3a), and then decompose it into four subdomains (Figure 3b) and solve the problem by the multi-domain model. The following field distribution is used as the exact solution:

$$\phi = x^3 + y^3 + z^3 - 3yx^2 - 3xz^2 - 3zy^2 \quad (26)$$



(a) Single domain model



(b) Four subdomains model

Figure 3. Modeling of the hollow cylinder.

Potential boundary conditions are imposed on the inner and outer cylindrical surfaces and normal flux boundary conditions on the top and bottom faces, according to equation (26). Comparative computations are performed on five pairs of nodal arrangements for the two models. The total numbers of degrees of freedom for these nodal arrangements are listed in the first and fifth columns in Table 2 for the multi- and single-domain models, respectively. In each pair of nodal arrangements, for comparability, we discretize the outer surfaces in both models with the same set of nodes. Table 2 shows that the numbers of degrees of freedom for the multi-domain model are slightly bigger than the single-domain model. This difference is due to the additional unknowns that are introduced into the overall problem by the subdomain interfaces in the multi-domain model.

The second and sixth columns in Table 2 list the CPU seconds for computing the equation coefficients by the single- and multi-domain models, respectively. The third and seventh columns indicate the time used for solving the overall system of equations. In the fourth and eighth

columns, the relative errors of nodal values of potential are presented. It is seen that, in all cases of nodal arrangement, the single-domain model used slightly less CPU seconds both for computing coefficients and for solving equations than its multi-domain counterpart, while the results are equally accurate. This observation is in contrast to that made for the conventional multi-domain BEM [11], where both the accuracy and efficiency can be dramatically improved by the multi-domain techniques in modeling slender objects. The reason for this may be that, in the FMM context, only the coefficients for pairs of nodes in the near field are directly computed, and thus the required floating-point operation counts to build the coefficient matrix and to solve the equation are of order $O(n)$ rather than $O(n^2)$ in the conventional multi-domain BEM.

Table 2. Results for the thick cylinder problem

Multi-domain model			
DOFs	T_{coef} (s)	T_{equ} (s)	err_ϕ
11888	711	1167	5.1×10^{-4}
47448	4127	6418	1.6×10^{-4}
106688	7753	12516	9.9×10^{-5}
189608	22482	34743	7.1×10^{-5}
296208	33889	65317	4.1×10^{-5}
Single-domain model			
DOFs	T_{coef} (s)	T_{equ} (s)	err_ϕ
10288	530	803	5.7×10^{-4}
41048	3906	5203	1.6×10^{-4}
92288	8065	8937	9.5×10^{-5}
164008	20297	22378	6.8×10^{-5}
256208	33373	43899	5.4×10^{-5}

5. Conclusions

The FMM techniques have been implemented in a multi-domain formulation of the HdBNM for numerical solution of Laplace's equation. The matrix-vector multiplication during the equation solution process is split into smaller scale ones at the subdomain level, and is accelerated by the FMM independently within individual subdomains.

Two numerical examples are presented to study the performance of the proposed method. High accuracy and efficiency have been demonstrated. It is clear that the method is suitable for analyzing large-scale multi-domain problems such as modeling of composites. In the conventional BEM, multi-domain strategies are usually used to get better computational efficiency for long slender objects. For the multi-domain FM-HdBNM, however, this is no longer feasible, because the FMM has already reduced the computational scale to nearly linear complexity.

The block diagonal preconditioner based on the leaves in the FMM tree structure may not be efficient for multi-domain formulation, because the coefficient matrix in the multi-domain equation is no longer diagonally dominated. Developing other forms of preconditioner in the FMM context, such as the sparse approximate inverse preconditioner [13], is necessary for performing large-scale computations of practical interest. This is an important subject of future research.

Acknowledgements

This work was supported by the CLUSTER of Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Belytchko T, Lu YY, Gu L. Element free Galerkin methods. *Int. J. Num. Meth. Engng.*, Vol. 37 (1994), pp. 229-256.
2. Atluri SN, Zhu T. A new meshless local Petrov-Galerkin approach in computational mechanics. *Computational Mechanics*, Vol. 22 (1998), pp. 117-127.
3. Mukherjee YX, Mukherjee S. The boundary node method for potential problems. *Int. J. Num. Meth. Engng.*, Vol. 40 (1997), pp. 797-815.
4. Zhang JM, Yao ZH, Li H. A hybrid boundary node method. *Int. J. Num. Meth. Engng.*, Vol, 53 (2002), pp. 751-763.
5. Zhang JM, Tanaka M, Matsumoto T. Meshless analysis of potential problems in three dimensions with the hybrid boundary node method. *Int. J. Num. Meth. Engng.*, Vol. 59 (2004), pp. 1147-1160.
6. Zhang JM, Tanaka M, Endo M. The hybrid boundary node method accelerated by fast multipole method for 3D potential problems. *Int. J. Num. Meth. Engng.*, Vol. 63 (2005), pp. 660-680.
7. Rokhlin V. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.*, Vol. 60 (1985), pp. 187-207.
8. Greengard L, Rokhlin V. A new version of the Fast Multipole Method for the Laplace equation in three dimensions. *Acta Numerica*, Vol. 6 (1997), pp. 229-269.
9. Nishida, T., and Hayami, K., Application of the fast multipole method to the 3D BEM analysis of electron guns, In Marchettia, M., Brebbia, C.A., and Aliabadi, M.H., Eds., *Boundary Elements XIX*, Computational Mechanics Publications (1997), pp. 613–622.
10. Yoshida K, Nishimura N, Kobayashi S. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *Int. J. Num. Meth. Engng.*, Vol. 50 (2001), pp. 525-547.
11. Kane JH. Boundary Element Analysis in Engineering Continuum Mechanics. Englewood Cliffs, NJ: Prentice-Hall, 1994.
12. Meric RA. Domain decomposition methods for Laplace's equation by the BEM. *Commun. Numer. Meth. Engng.*, Vol. 16 (2000), pp. 545-557.
13. Benzi M, Tuma M. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, Vol. 19 (1998), pp. 968-994.