Transactions of JASCOME, Vol. 4 (July 2004), Paper No. 04-070910 JASCOME

A TREE DATA STRUCTURE FOR MLS APPROXIMATION OF BOUNDARY VARIABLES IN HYBRID BNM

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

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

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

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

By coupling the moving least squares (MLS) approximation with a modified functional, the hybrid boundary node method (Hybrid BNM) is a boundary only, truly meshless, method. Like the conventional BEM, the unsymmetrical and dense coefficient matrix limits its application to small-scale problems. Recently, the Hybrid BNM has been combined with the fast multipole method (FMM). In the combined approach, however, the MLS approximation on a surface with a large number of nodes distributed appears to be another bottleneck for large-scale computation. In this paper, the tree data structure, used in the hierarchical decomposition of the domain in FMM, is adapted and applied to accelerate the MLS approximation. Formulations and algorithm are given and a numerical example presented to demonstrate the efficiency of the proposed approach.

Keywords: Moving least squares, Fast multipole method, Hybrid BNM, Tree data structure

1. Introduction

In numerical solutions of engineering problems, two main difficulties usually arise. One is the discretization of the geometry and another computational scale. 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-labor cost of introducing geometric meshes in complex-shaped domains. Many kinds of meshless methods have been proposed so far. The hybrid boundary node method (Hybrid BNM), suggested by Zhang et al. [1], which combines the MLS interpolation scheme with the hybrid variational formulation, not only has the dimensionality advantage of BEM, but also is a truly meshless method, therefore, substantially simplifies the discretization task. However, like the traditional BEM, its system matrix is dense and unsymmetrical, and demands $O(N^2)$ memory and $O(N^3)$ operations. In order to reduce the computational complexity and memory requirement, we have combined the Hybrid BNM with the fast multipole method (FMM) [2, 3].

In the combined approach [4] (here called FM-HBNM), 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 FMM. An oct-tree data structure is used to hierarchically subdivide the domain into well-separated cells. Only the coefficients for near nodes are explicitly computed and stored. The influences of the nodes far from the observation point are approximated using multipole expansions. As a result, both the memory and CPU time required for solving the set of equations are reduced. However, unlike BEM using element-based interpolation functions to represent the approximated solutions of boundary variables, Hybrid BNM uses moving least squares (MLS) approximation. The MLS approximation can be costly and may lead to a serious exhaustion of the computer memory. In order to overcome this shortcoming, we use a binary tree data structure, similar to the oct-tree data structure used in FMM, to speed up the MLS approximation. In this paper, we will focus on evaluation of the MLS shape functions using the binary tree data structure, while we will first give brief descriptions of Hybrid BNM and FM-HBNM.

2. The Hybrid BNM and FM-HBNM

The Hybrid BNM bases on a modified variational principle. Taking 3-D potential problems as an example (for detailed discussion see [1]), the functions in the modified variational principle considered 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

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

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

Suppose that N nodes are distributed on the bounding surface of the domain, the potential field within the domain is approximated using fundamental solutions as follows:

$$u = \sum_{I=1}^{N} u_I^s x_I \quad (2)$$

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

$$q = \sum_{I=1}^{N} \frac{\partial u_I^s}{\partial n} x_I \tag{3}$$

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

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

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

The boundary potential field $\tilde{u}$ and boundary normal flux $\tilde{q}$ are interpolated by the MLS [1].

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

$$\tilde{q}(\mathbf{s}) = \sum_{I=1}^{N} \Phi_I(\mathbf{s}) \hat{q}_I \tag{6}$$

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

Taking the local sub-domain around each node into account, the stationary conditions can be obtained by taking variations in Eq. (1) with respect to the independent variables. This gives the following set of equations:

$$\mathbf{U}\mathbf{x} = \mathbf{H}\hat{\mathbf{u}} \tag{7}$$

$$\mathbf{V}\mathbf{x} = \mathbf{H}\hat{\mathbf{q}} \tag{8}$$

where **U**, **V** and **H** are defined as:

$$U_{IJ} = \int_{\Gamma_s^J} u_I^s v_J(Q) d\Gamma \tag{9}$$

$$V_{IJ} = \int_{\Gamma_s^J} \frac{\partial u_I^s}{\partial n} v_J(Q) d\Gamma \tag{10}$$

$$H_{IJ} = \int_{\Gamma_s^J} \Phi_I(\mathbf{s}) v_J(Q) d\Gamma \tag{11}$$

where v_J is a weight function and $\mathbf{s}$ 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. Therefore, the integrals in Eqs. (9), (10) and (11) can be calculated without using boundary elements (for details refer to [1]).

For a well-posed problem, either $\hat{u}_I$ or $\hat{q}_I$ is known at a node $\mathbf{s}_I$ on the boundary, thus Eqs. (7) and (8) can be solved for unknown parameters **x**. Then, by back-substitution into Eqs. (7) and (8), the boundary unknowns are obtained for both potentials and normal fluxes by solving Eqs. (7) and (8) with **H** being the coefficient matrix.

The coefficient matrices **U** and **V** are dense and unsymmetrical. It requires $O(N^2)$ memory to store them and $O(N^3)$ CPU time to solve them if a direct solver is employed. If we use an iterative solver, such as GMRES, the N^2 cost of forming the dense matrix-vector product in the system of equations will dominate the total cost. In the case of Eqs. (7) and (8), the matrix-vector product is equivalent to evaluating the potentials or their derivates at N nodes, using Eqs. (2) and (3). Therefore, it is possible to reduce the cost of GMRES by accelerating the potential calculation.

In FM-HBNM, we use a constructed hierarchy of boxes to refine the computational domain into smaller and smaller regions. At refinement level 0, we have the entire computational domain. Refinement level l+1 is obtained recursively from level l by subdivision of each into eight equal parts. This yields a tree structure, where the eight boxes at level l+1 obtained by subdivision of a box at level l are considered its children. We stop the box subdivision if the number of nodes included in the box is smaller than a given value. If a child box contains no node, we delete it. We call a childless box a *leaf* and two boxes *neighbors* if they are at the same level and share at least a vertex.

Given evaluation point included in a leaf and using the tree data structure described above, we can divide sum (2) and (3) into two parts. Part 1 is the sum of the contributions of the nodes contained in the neighbors of the leaf (these nodes called *near nodes*), and part 2 that of the nodes that are outside all the neighbors (these nodes called *far nodes*). We compute the sum for the near nodes directly, while do the summation for the far nodes by means of fast multipole expansions at a cost proportional to N at each iteration step (for details see [4]). Since the coefficients in matrices **U** and **V** are explicitly computed and stored for near nodes, only, and the computational cost is proportional to N, the overall complexity for solving Eqs. (7) and (8) is of order N.

From the above discussion, we can see that FM-HBNM only concerns matrices **U** and **V**, while leaves matrix **H** intact. There are two usages of **H** in Hybrid BNM. One is computing the right hand side vector of Eqs. (7) and (8), while the other is solving the boundary unknowns $\hat{\mathbf{u}}$ and $\hat{\mathbf{q}}$ by Eqs. (7) and (8) after **x** has been solved. Since the MLS approximation in Hybrid BNM is conducted on individual panels separately, the matrix **H**, unlike **U** and **V**, is diagonally blocked. Even so, when a panel with a large number of nodes located, the size of the corresponding block may be extremely large, and the evaluation of the shape functions in Eq. (11) can be expensive. In order to circumvent this problem, we use a binary tree data structure to speed up MLS approximation and reduce the memory requirements for storing matrix **H**.

3. Original MLS algorithm

In Hybrid BNM, the MLS approximation is required on the bounding surface, only, as the nodes lie only on the boundary of a 3-D body. It is assumed that the bounding surface of a 3-D body is a union of piecewise smooth segments. We call these segments *panels*, and perform MLS approximation on each panel separately.

In Reference [1], we have proposed a general MLS approximation algorithm on an arbitrary panel. For a panel over which randomly located a number of nodes $\{\mathbf{s}^I\}$, I=1, 2, ..., n, the MLS interpolants for a boundary variable $f(\mathbf{s})$ is defined by

$$f(\mathbf{s}) = \sum_{j=1}^{m} p_j(\mathbf{s}) a_j(\mathbf{s}) = \mathbf{p}^{\mathrm{T}}(\mathbf{s}) \mathbf{a}(\mathbf{s}) \tag{12}$$

where $\mathbf{s}$ is a field point with parametric coordinates (s_1, s_2), defined in the range [0, 1]; and $p_j(\mathbf{s})$, j=1, 2, ..., m are monomials in (s_1, s_2). The monomials $p_j(\mathbf{s})$ provide the intrinsic polynomial bases for $f(\mathbf{s})$. In the study, a quadratic background basis is used, i.e.

$$\mathbf{p}^{\mathrm{T}}(\mathbf{s}) = [1, s_1, s_2, s_1^2, s_1 s_2, s_2^2], \quad m = 6 \tag{13}$$

The coefficient vectors $\mathbf{a}(\mathbf{s})$ and $\mathbf{b}(\mathbf{s})$ are determined by minimizing a weighted discrete L_2 norm, defined as

$$J(\mathbf{s}) = \sum_{I=1}^{n} w_I(\mathbf{s}) \left[\mathbf{p}^{\mathrm{T}}(\mathbf{s}^I)\mathbf{a}(\mathbf{s}) - \hat{f}_I \right]^2 \tag{14}$$

where $w_I(\mathbf{s})$ is a weight function corresponding to node $\mathbf{s}^I$ and $\hat{f}_I$ is the nodal value.

Solving for $\mathbf{a}(\mathbf{s})$ and $\mathbf{b}(\mathbf{s})$ by minimizing J in Eq. (14), and substituting them into Eq. (12) gives a relation which can be written in the forms with interpolation functions similar to those used in FEM, as follows:

$$\tilde{f}(\mathbf{s}) = \sum_{I=1}^{n} \Phi_I(\mathbf{s}) \hat{f}_I \tag{15}$$

where the shape functions is expressed as

$$\Phi_I(\mathbf{s}) = \sum_{j=1}^{m} p_j(\mathbf{s}) \left[A^{-1}(\mathbf{s}) B(\mathbf{s}) \right]_{jI} \tag{16}$$

with matrices $A(\mathbf{s})$ and $B(\mathbf{s})$ defined by

$$A(\mathbf{s}) = \sum_{I=1}^{n} w_I(\mathbf{s}) \mathbf{p}(\mathbf{s}^I) \mathbf{p}^{\mathrm{T}}(\mathbf{s}^I) \tag{17}$$

and

$$B(\mathbf{s}) = \left[w_1(\mathbf{s})\mathbf{p}(\mathbf{s}^1), w_2(\mathbf{s})\mathbf{p}(\mathbf{s}^2), \cdots, w_n(\mathbf{s})\mathbf{p}(\mathbf{s}^n) \right] \tag{18}$$

The MLS approximation is well-defined only when the matrix $A(\mathbf{s})$ in Eq. (17) is non-singular.

Choosing a proper weight function is an important aspect in a successful implementation of MLS approximation. The choice of weight functions and the consequences of a choice are discussed in detail elsewhere [5]. In the study, we use Gaussian weight function. The Gaussian weight function corresponding to a node $\mathbf{s}^I$ can be written by

$$w_I(\mathbf{s}) = \begin{cases} \dfrac{\exp[-(d_I/c_I)^2] - \exp[-(\hat{d}_I/c_I)^2]}{1 - \exp[-(\hat{d}_I/c_I)^2]}, & 0 \le d_I \le \hat{d}_I \\ 0, & d_I \ge \hat{d}_I \end{cases} \tag{19}$$

where c_I is a constant controlling the shape of the weight function, and $\hat{d}_I$ is the size of the support for the weight function w_I. It can be seen from the above equation that the weight function has a compact support determined by the parameter $\hat{d}_I$. 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 $\hat{d}_I$ being the half-length of major axis of the ellipse. Denoting the half-length of minor axis by $\hat{d}'_I$, we have the following expression for d_I:

$$d_I = \sqrt{(s_1 - s_1^I)^2 + \frac{\hat{d}_I^2}{\hat{d}_I'^2}(s_2 - s_2^I)^2} \tag{20}$$

From Eqs. (16) and (18), it is seen that $\Phi_I(\mathbf{s}) = 0$ in case $w_I(\mathbf{s})=0$. The fact that $\Phi_I(\mathbf{s})$ vanishes for $\mathbf{s}$ not in the support of node $\mathbf{s}^I$ preserves the local character of the MLS approximation. In order to retain the local character, we should use small values for $\hat{d}_I$ and $\hat{d}'_I$. On the other hand, however, to ensure the regularity of $\mathbf{A}(\mathbf{s})$, $\hat{d}_I$ and $\hat{d}'_I$ should be chosen in such a way that they are large enough to have a sufficient number of nodes to be covered in the domain of definition of every sample point. In this study, we choose $\hat{d}_I$ and $\hat{d}'_I$ such that $4m \sim 8m$ nodes are included in the support of a node.

At a panel, we compute the shape functions according to the following routine:

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
   - determine the nodes $\mathbf{s}^I$ that $w_I(\mathbf{s})>0$;
   - calculate the right hand side of Eq. (17);
   - 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 Eqs. (18) and (16).

4. Binary tree data structure for MLS approximation

In BEM, the number of shape functions for an evaluation point equals to the number of nodes of an element, while in Hybrid BNM, the number of MLS shape functions equals to the total number of nodes on the panel. Although most of the MLS shape functions equal to zero due to the compact support of the weight function of each node, we cannot determine before computation which of them are zero. The reason is, in the input data structure in Hybrid BNM, there is no information of connectivity between the nodes. This leads to two drawbacks in the original MLS approximation algorithm:

1. Evaluation of the shape functions for one field 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 total number of nodes is very large.
2. The locations of the non-zero entries in every row of matrix $\mathbf{H}$ (see Eq. (11)) 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 may lead to an exhaustion of computer memory.

In order to overcome the 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 3-D problems is carried out on 2-D 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 following parameters:

- *Center.s1* denotes the value of parametric coordinate of the center of the cell in s_1 direction.
- *Center.s2* denotes the value of parametric coordinate of the center of the cell in s_2 direction.
- *H.s1* denotes the side length of the cell in s_1 direction.
- *H.s2* denotes the side length of the cell in s_2 direction.

- *Dmax.s1* denotes the maximum value of $\hat{d}_I$ among the nodes included in the cell.
- *Dmax.s2* 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.s1* is bigger than *Dmax.s1*, we subdivide the cell S into two equal cells in s_1 direction; and if *H.s2* is bigger than *Dmax.s2*, we subdivide the cell S into two equal cells in s_2 direction. 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 *neighbor list* for each leaf. Taking a leaf L into account and looping over all other leaves L_i, we consider L_i to be L's neighbor and add it to L's neighbor list, if the distances between their centers in both s_1 and s_2 directions are smaller than *Dmax.s1* and *Dmax.s2* associated with L_i, respectively. A leaf is also a neighbor of itself. Now, instead of creating an $n \times n$ square block of **H**, we associate each leaf a $j \times k$ sub-matrix $\mathbf{h}_{jk}$, where j denotes the number of nodes included in the leaf; and k denotes the number of nodes included in the neighbors of the leaf. This scheme saves the memory considerably.

With the binary tree data structure, the routine for computing the shape functions changes to the following steps.

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 **s**.
5. Loop over the nodes included in the neighborhood of L_c
   - determine the nodes $\mathbf{s}^I$ that $w_I(\mathbf{s})>0$;
   - calculate the right hand side of Eq. (17);
   - add contributions to **A**(**s**).
6. Solve the inversion of **A**(**s**).
7. Loop over all the nodes included in the neighborhood 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 Eqs. (18) and (16).

In the above algorithm, the loop for checking the weight functions contains only the nodes that are in the neighborhood of a leaf. When the total number of nodes located at the panel is large, the CPU time saved by the new algorithm will be obvious.

5. Numerical results

The proposed technique has been implemented in a computer code written in C++, and tested with a cube. Computations for a variety of number of nodes uniformly scattered on the faces are performed on a desktop computer with an Intel Celeron CPU (2.40GHz). Following Reference [5], the support sizes of the weight function, $\hat{d}_I$ and $\hat{d}'_I$ in Eq. (20), are chosen to be $4.0h_1$ and $4.0h_2$, with h_1 and h_2 being the minimum distances between the neighbouring nodes in s_1 and s_2 directions, respectively. For comparison, the models have also been calculated using the original FM-HBNM in the cases where it is capable of solving them. The CPU time (in second) and memory sizes (in MB) for computing and storing matrix **H** required by the new and original algorithm, respectively, are presented in Figure 1. From Figure 1, it is seen that the new algorithm is much faster than the original algorithm and uses less memory. Computations by means of the original algorithm are restricted to 21600 nodes (due to hardware limitation), while the new algorithm is capable of solving problems with the total number of degrees of freedom up to 150000.

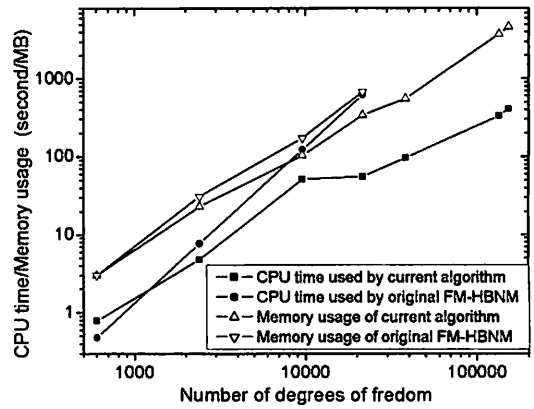

Figure 1 Memory usage and CPU timing results

Conclusions

This paper presents an enhanced implementation of MLS approximation of boundary variables with a binary tree data structure. The new implementation decreases the execution time with lower memory requirement, thus significantly increases the size of problem that can be solved within available computer resources.

The new algorithm can be exploited in any meshless method that involves MLS approximation.

Acknowledgements

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

References

1. Zhang, J.M., Tanaka, Masa., Matsumoto, T., Meshless analysis of potential problems in three dimensions with the hybrid boundary node method, Int. J. Num. Meth. Eng., Vol. 59 (2004), pp. 1147-1160.
2. 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.
3. 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. Eng., Vol. 50 (2001), pp. 525-547.
4. Zhang J.M., Tanaka Masa., Endo M., The hybrid boundary node method accelerated by fast multipole method for 3D potential problems. Int. J. Num. Meth. Eng., to be submitted (2004).
5. Atluri S.N., Zhu T., A new meshless local Petrov-Galerkin approach in computational mechanics. Computational Mechanics, Vol. 22 (1998), pp. 117-127.