A Tree Data Structure for MLS Approximation of Boundary Variables in the Hybrid BNM

Masa. Tanaka, J. Zhang, M. Endo

Shinshu University
Faculty of Engineering

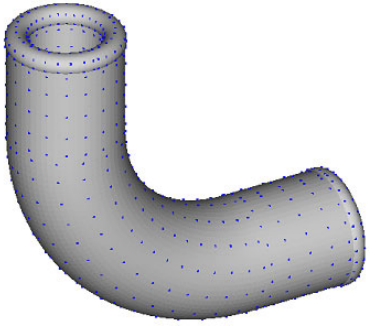
Outline

- Hybrid BNM and FM-HBNM
- Original MLS algorithm
- MLS with a binary tree data structure
- Numerical results
- Conclusions

Hybrid BNM and FM-HBNM

Main features:

- Combines modified functional with the *Moving Least Squares* (MLS) approximation
- Boundary-only truly meshless method
- Three independent variables
  - internal temperature u
  - boundary temperature $\tilde{u}$
  - boundary normal flux $\tilde{q}$



Example of meshless discretization

Hybrid BNM and FM-HBNM(2)

➢Variables approximation

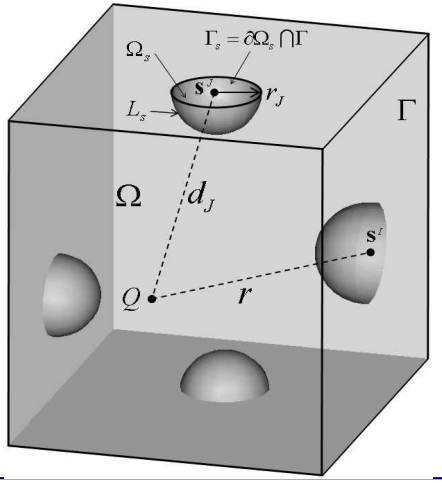
- Domain variables

$$u = \sum_{I=1}^{N} u_I^s x_I \qquad u_I^s = \frac{1}{\kappa}\frac{1}{4\pi r(Q,\mathbf{s}_I)}$$

- Boundary variables

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

➢Local weak form



$$\int_\Gamma (q-\tilde{q})\delta u d\Gamma - \int_\Omega u_{,ii}\delta u d\Omega + \int_{\Gamma_q} (\tilde{q}-\bar{q})\delta\tilde{u} d\Gamma - \int_\Gamma (u-\tilde{u})\delta\tilde{q} d\Gamma = 0$$

$$\sum_{I=1}^{n}\int_{\Gamma_s} \frac{\partial u_I^s}{\partial n} v_J(Q)x_I d\Gamma = \sum_{I=1}^{n}\int_{\Gamma_s} \Phi_I(\mathbf{s})v_J(Q)\hat{q}_I d\Gamma$$

$$\sum_{I=1}^{n}\int_{\Gamma_s} u_I^s v_J(Q)x_I d\Gamma = \sum_{I=1}^{n}\int_{\Gamma_s} \Phi_I(\mathbf{s})v_J(Q)\hat{\phi}_I d\Gamma$$

Hybrid BNM and FM-HBNM(4)

➢System of equations – final form

$$\mathbf{Ux} = \mathbf{H\hat{q}}$$

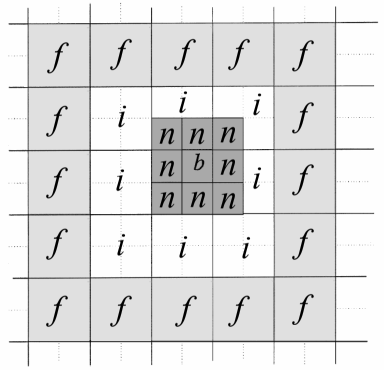
$$\mathbf{Vx} = \mathbf{H\hat{u}}$$

where

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

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

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



f: far field

i: interaction list

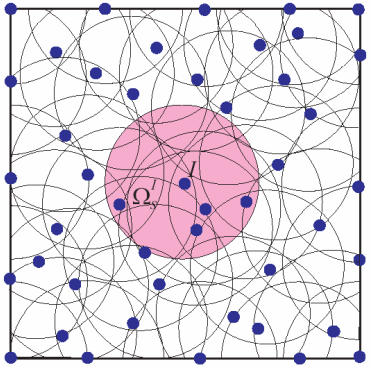
n: neighbor

$$\sum_{I=1}^{N} \int_{\Gamma_J} u_I^s v_J(Q) x_I' d\Gamma = \overset{Near\ nodes}{\sum_{I}} \int_{\Gamma_J} u_I^s v_J(Q) x_I' d\Gamma + \overset{Far\ nodes}{\sum_{I}} \int_{\Gamma_J} u_I^s v_J(Q) x_I' d\Gamma$$

Near nodes ~ n *Far nodes* ~ $i + f$

Original MLS algorithm

- Parametric unit square



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

$$\tilde{q}(\mathbf{s}) = \sum_{j=1}^{m} p_j(\mathbf{s}) b_j(\mathbf{s}) = \mathbf{p}^{\mathrm{T}}(\mathbf{s})\mathbf{b}(\mathbf{s})$$

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

- weighted discrete norms

$$J_1(\mathbf{s}) = \sum_{I=1}^{N} w_I(\mathbf{s}) \left[\mathbf{p}^{\mathrm{T}}(\mathbf{s}^I)\mathbf{a}(\mathbf{s}) - \hat{u}_I \right]^2$$

$$J_2(\mathbf{s}) = \sum_{I=1}^{N} w_I(\mathbf{s}) \left[\mathbf{p}^{\mathrm{T}}(\mathbf{s}^I)\mathbf{b}(\mathbf{s}) - \hat{q}_I \right]^2$$

Where $w_I(\mathbf{s})$ is a weight function with compact support

$\hat{u}_I$ and $\hat{q}_I$ are nodal values

Original MLS algorithm (2)

- Solving for $\mathbf{a}(\mathbf{s})$ and $\mathbf{b}(\mathbf{s})$ by minimizing J_1 and J_2

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

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

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

where

$$A(\mathbf{s}) = \sum_{I=1}^{N} w_I(\mathbf{s})\mathbf{p}(\mathbf{s}^I)\mathbf{p}^{\mathbf{T}}(\mathbf{s}^I)$$

$$B(\mathbf{s}) = \left[w_1(\mathbf{s})\mathbf{p}(\mathbf{s}^1), w_2(\mathbf{s})\mathbf{p}(\mathbf{s}^2), \cdots, w_N(\mathbf{s})\mathbf{p}(\mathbf{s}^N)\right]$$

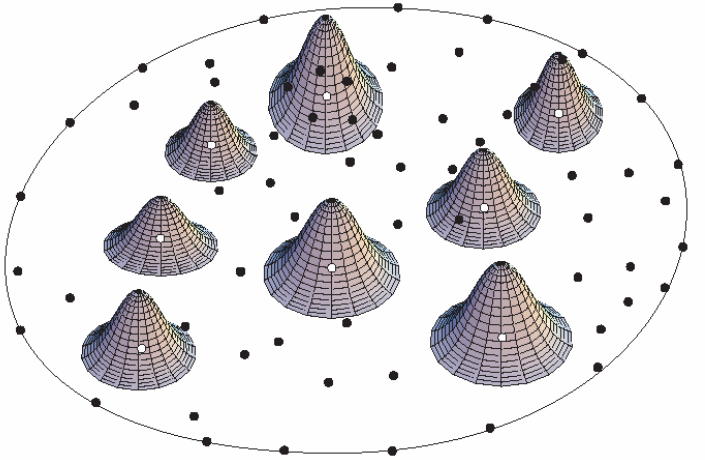
- Local character

$$w_I(\mathbf{s}) = 0 \rightarrow B(\mathbf{s}) = 0 \rightarrow \Phi_I(\mathbf{s}) = 0$$

Original MLS algorithm (3)

- Weight function

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



Shape of weight function

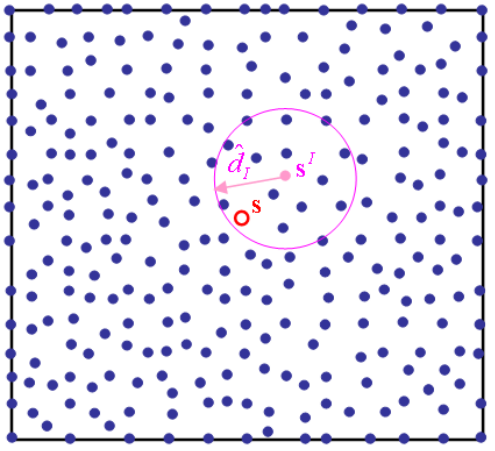
Original MLS algorithm (4)

- Algorithm

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

$$B(\mathbf{s}) = \left[w_1(\mathbf{s})\mathbf{p}(\mathbf{s}^1), w_2(\mathbf{s})\mathbf{p}(\mathbf{s}^2), \cdots, w_N(\mathbf{s})\mathbf{p}(\mathbf{s}^N) \right]$$

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



n nodes on a panel

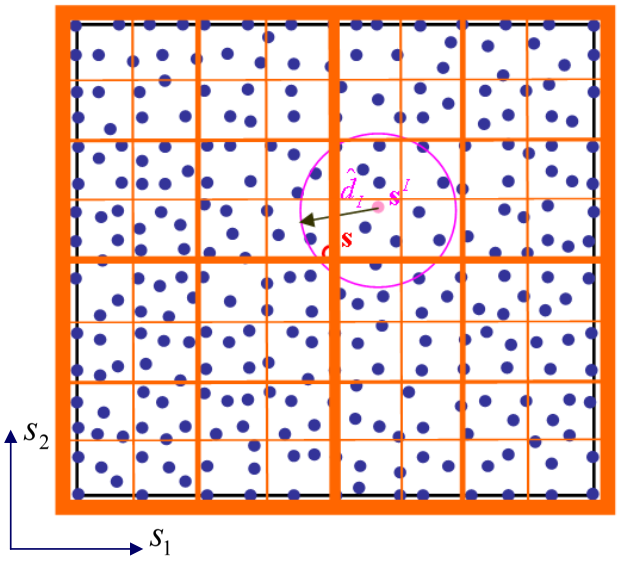
1. Loop over all nodes located on the panel
   - determine the nodes $\mathbf{s}^I$ that $w_I(\mathbf{s}) > 0$;
   - calculate $w_I(\mathbf{s})\mathbf{p}(\mathbf{s}^I)\mathbf{p}^{\mathrm{T}}(\mathbf{s}^I)$;
   - add $w_I(\mathbf{s})\mathbf{p}(\mathbf{s}^I)\mathbf{p}^{\mathrm{T}}(\mathbf{s}^I)$ to A(s).
2. Solve the inversion of A(s).
3. 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 $\Phi_I(\mathbf{s})$.

Remarks:

1. Loop has to be over all the nodes on the panel.
2. Size of matrix H ($H_{IJ} = \int_{\Gamma_s^J} \Phi_I(\mathbf{s}) v_J(Q) d\Gamma$) is $n \times n$.

MLS with a binary tree

- Panel partitioning with cells



With a cell associate:

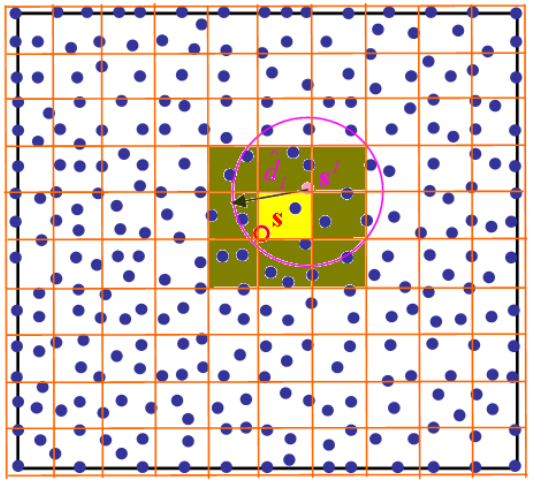
1. *H.s1*, *H.s2* denotes the side length of the cell in s_1 and s_2 direction, respectively.
2. *Dmax.s1*, *Dmax.s2* denotes the maximum value of $\hat{d}_I$ for the nodes included in the cell in s_1 and s_2 direction, respectively.

Build the binary tree

1. Let the root cell contains the entire panel.
2. Recursively repeat:
   if *H.s1* > *Dmax.s1*, subdivide a cell into two equal cells in s_1 direction;
   if *H.s2* > *Dmax.s2*, subdivide a cell into two equal cells in s_2 direction.
3. Refer the cells at the finest level as *leaves*. Create a *neighbor list* for each leaf.
4. Associate each leaf a $j \times k$ sub-matrix h_{jk}.
   j denotes the number of nodes included in the leaf, and *k* the number of nodes in the neighborhood.

MLS with a binary tree (2)

- New algorithm



1. Create the binary tree data structure.
2. Find the leaf that includes the evaluation point.
3. Loop over the nodes included in the neighborhood
   - determine the nodes s^I that $w_I(s) > 0$;
   - calculate $w_I(s)p(s^I)p^T(s^I)$;
   - add $w_I(s)p(s^I)p^T(s^I)$ to A(s).
4. Solve the inversion of A(s).
5. Loop over the nodes included in the neighborhood. For each node s^I that $w_I(s) > 0$, calculate $w_I(s)p(s^I)$ and then $\Phi_I(s)$.

$$A(\mathbf{s}) = \sum_{I \in \text{nodes in Neighborhood}} w_I(\mathbf{s})\mathbf{p}(\mathbf{s}^I)\mathbf{p}^{\mathrm{T}}(\mathbf{s}^I)$$

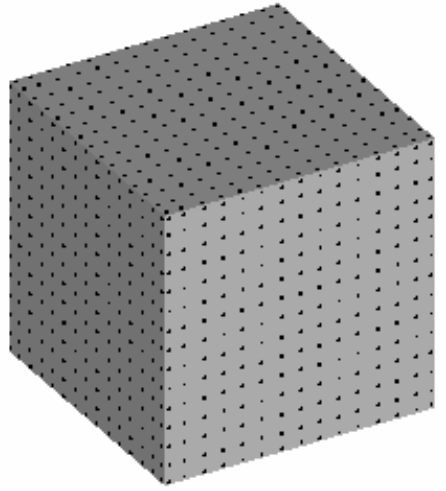
$$B(\mathbf{s}) = \left[w_1(\mathbf{s})\mathbf{p}(\mathbf{s}^1), w_2(\mathbf{s})\mathbf{p}(\mathbf{s}^2), \cdots \right]$$

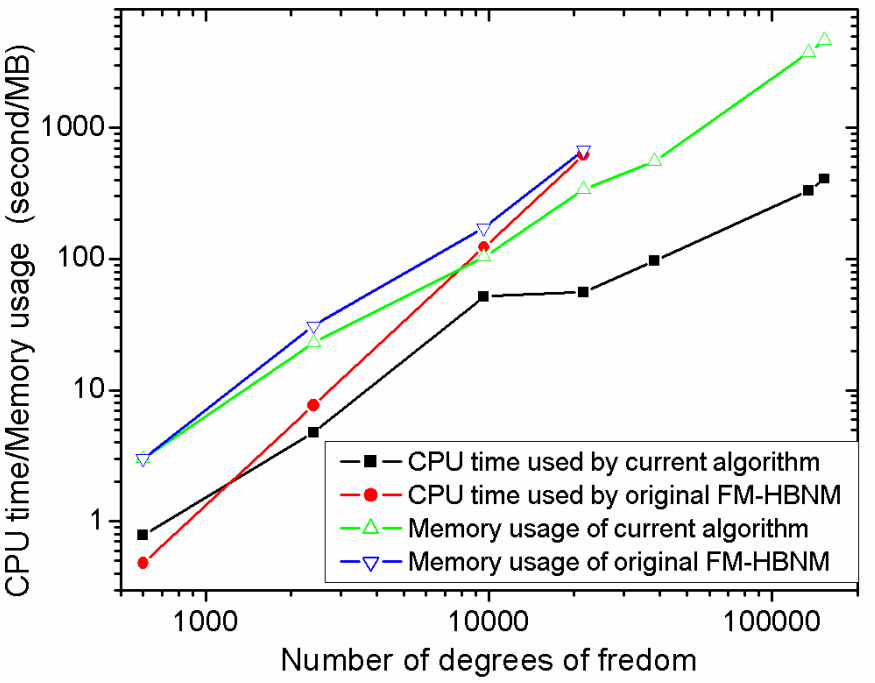
$$\Phi_I(\mathbf{s}) = \sum_{j=1}^{m} p_j(\mathbf{s}) \left[A^{-1}(\mathbf{s}) B(\mathbf{s}) \right]_{jI}$$

Remarks:

1. Loop is restricted to the neighboring nodes.
2. Matrix H is replaced by a number of sub-matrices h_{jk}.

Numerical example





Memory usage and CPU timing results for computing matrix H

Conclusions

- We present an enhanced implementation of MLS approximation 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.