Adaptive spatial decomposition in Fast Multipole Method

Jianming Zhang, Masataka Tanaka

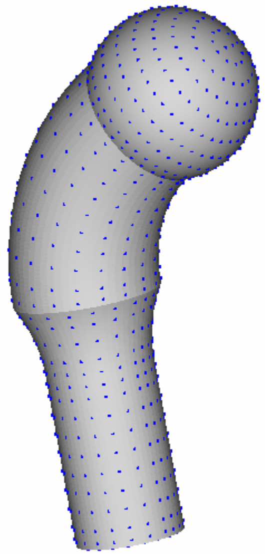
Nov. 15, 2006

Shinshu University
Faculty of Engineering

Hybrid BNM

Main features:

- Combines a modified functional with the *Moving Least Squares* (MLS) approximation
- Three independent variables
  - internal temperature
  - boundary temperature
  - boundary normal flux



Example of meshless discretization

Variables approximation

- Domain variables

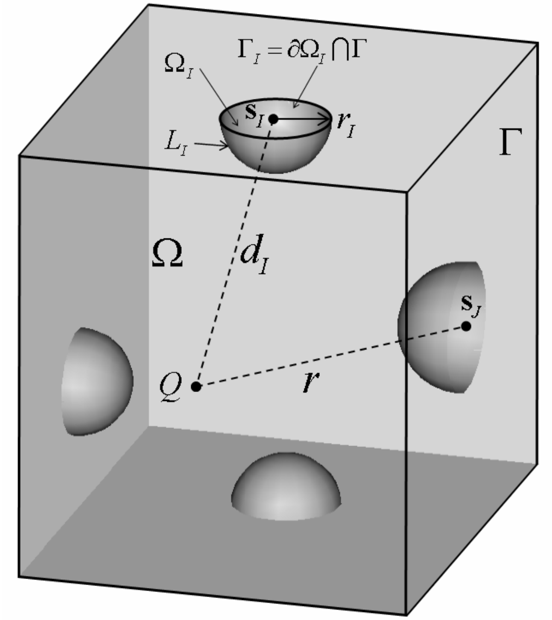
$$\phi = \sum_{I=1}^{N} \phi_I^s x_I \qquad \phi_I^s = \frac{1}{\kappa} \frac{1}{4\pi r(Q, \mathbf{s}_I)}$$

- Boundary variables

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

Hybrid BNM (2)

➢System of equations



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

$$\boxed{\mathbf{Vx} = \mathbf{H}\hat{\boldsymbol{\phi}}}$$

$$U_{IJ} = \int_{\Gamma_s^J} \phi_I^s v_J(Q)\, d\Gamma$$

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

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

- Three purposes of elements in BEM:
  - To interpolate Boundary variables
  - To facilitate boundary integration
  - To approximate the geometry

Iterative solver

➢ **Matrix-vector multiplication**

$$x_I'^{k+1} = \sum_{J=1}^{N} \int_{\Gamma_I} \phi_J^s v_I(Q) x_J^k d\Gamma$$

$$x_I'^{k+1} = \sum_{J=1}^{N} \int_{\Gamma_I} \frac{\partial \phi_J^s}{\partial n} v_I(Q) x_J^k d\Gamma$$

Complexity of an iterative solver:

Memory: $O(N^2)$ **CPU time: $O(N^2)$**

Fast Multipole Method (FMM):

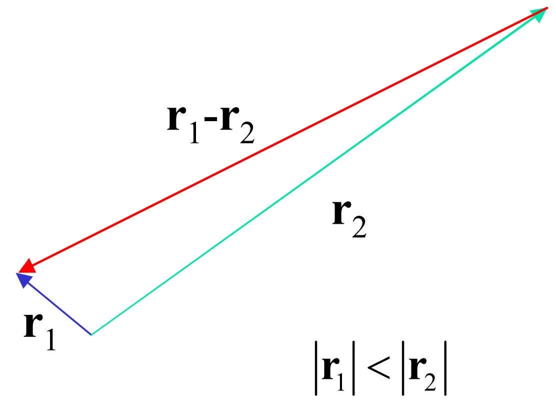
Reducing computational complexity to $O(N)$

Addition theorems

➢ First addition theorem

Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be two vectors with spherical coordinates (r_1, α_1, β_1) and (r_2, α_2, β_2), respectively. It follows

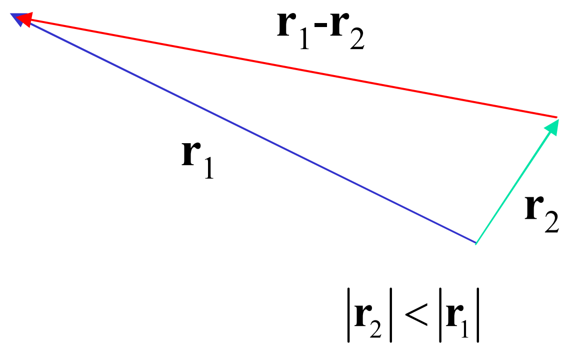
$$\frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|} = \begin{cases} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} R_n^m(\mathbf{r}_1) \overline{S_n^m(\mathbf{r}_2)}, & |\mathbf{r}_1| < |\mathbf{r}_2| \\ \sum_{n=0}^{\infty} \sum_{m=-n}^{n} R_n^m(\mathbf{r}_2) \overline{S_n^m(\mathbf{r}_1)}, & |\mathbf{r}_1| > |\mathbf{r}_2| \end{cases}$$



where

$$R_n^m(\mathbf{r}) = \frac{1}{(n+m)!} P_n^m(\cos\alpha) e^{im\beta} r^n$$

$$S_n^m(\mathbf{r}) = (n-m)! P_n^m(\cos\alpha) e^{im\beta} \frac{1}{r^{n+1}}$$

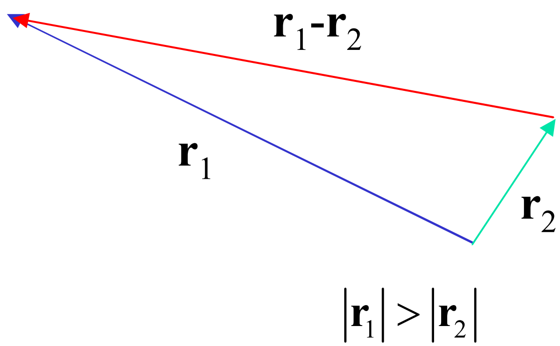


Addition theorems (2)

➢Second addition theorem

Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be two vectors such that $|\mathbf{r}_1|>|\mathbf{r}_2|$, then

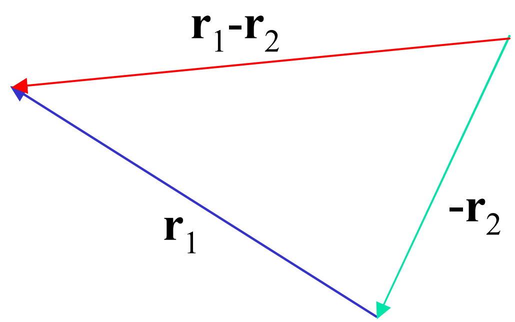
$$S_n^m(\mathbf{r}_1-\mathbf{r}_2)=\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}\overline{R_{n'}^{m'}(\mathbf{r}_2)}S_{n+n'}^{m+m'}(\mathbf{r}_1)$$



➢Third addition theorem

Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be two arbitrary vectors, then

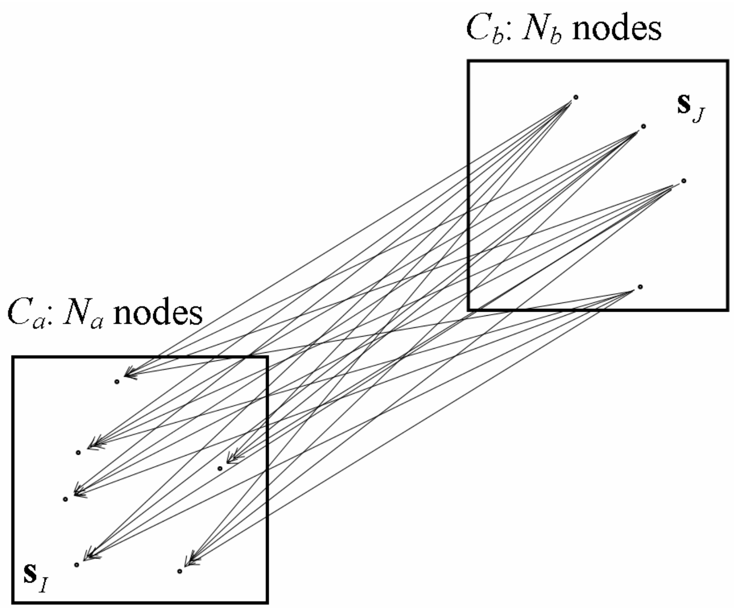
$$R_n^m(\mathbf{r}_1-\mathbf{r}_2)=\sum_{n'=0}^{n}\sum_{m'=-n'}^{n'}R_{n'}^{m'}(-\mathbf{r}_2)R_{n-n'}^{m-m'}(\mathbf{r}_1)$$



Fast multipole

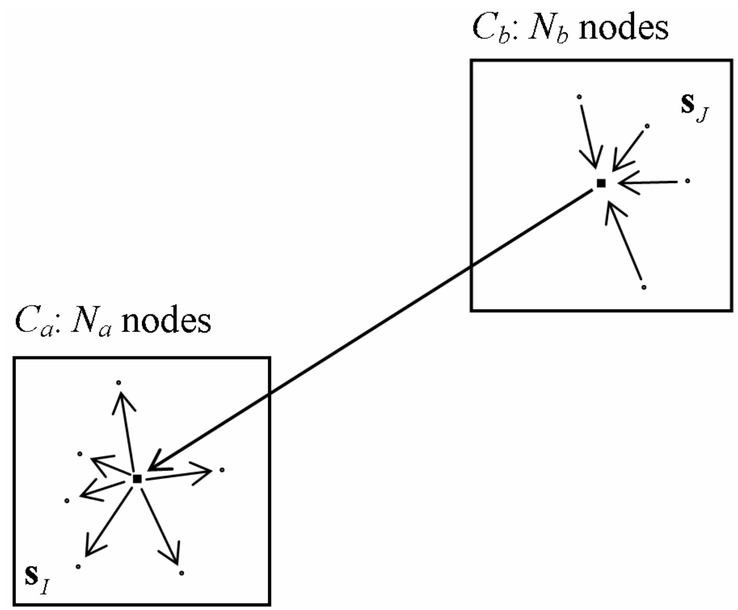
- **Ideas of FMM**

Node-node interactions



Complexity $O(N_a N_b)$

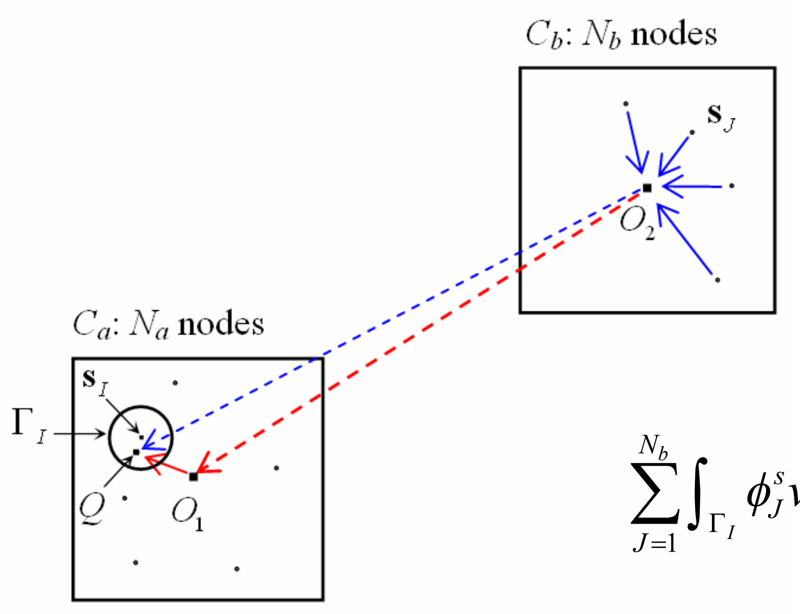
Cell-cell interactions



Complexity $O(N_a + N_b)$

Fast multipole (2)

- **Multipole expansion**



$$\phi_J^s = \frac{1}{4\pi\kappa}\frac{1}{r(Q,\mathbf{s}_J)} = \frac{1}{4\pi\kappa}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\boxed{\overline{S_n^m}(\overrightarrow{O_2Q})}\,\boxed{R_n^m(\overrightarrow{O_2\mathbf{s}_J})}$$

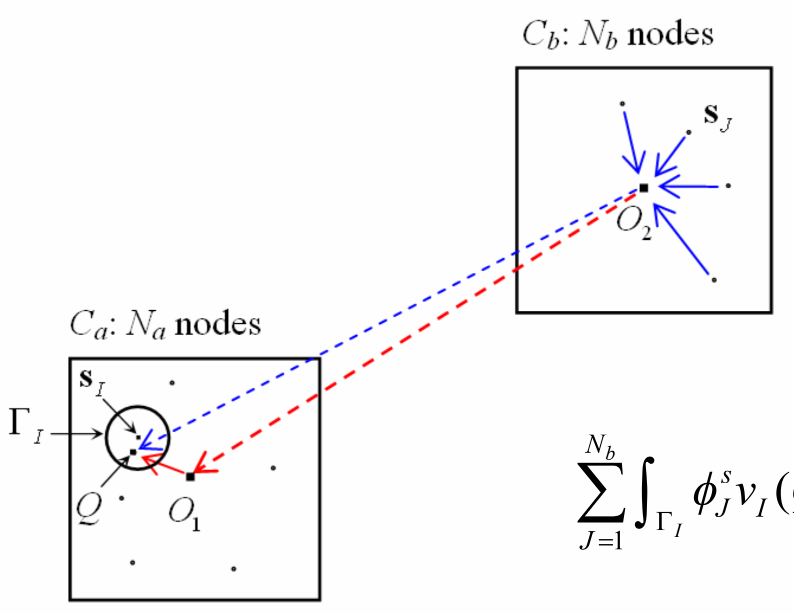
for $\quad |\overrightarrow{O_2Q}| > |\overrightarrow{O_2\mathbf{s}_J}|$

$$\sum_{J=1}^{N_b}\int_{\Gamma_I}\phi_J^s v_I(Q)x_J' d\Gamma = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\int_{\Gamma_I}\frac{1}{4\pi\kappa}\boxed{\overline{S_n^m}(\overrightarrow{O_2Q})}\,v_I(Q)d\Gamma\,\boxed{M_n^m(O_2)}$$

where $\quad \boxed{M_n^m(O_2) = \sum_{J=1}^{N_b} R_n^m(\overrightarrow{O_2\mathbf{s}_J})x_J'}$

Fast multipole (3)

➢ Local expansion



$$\overline{S_n^m(\overrightarrow{O_2Q})} = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} (-1)^{n'} R_{n'}^{m'}(\overrightarrow{O_1Q}) \overline{S_{n+n'}^{m+m'}(\overrightarrow{O_1O_2})}$$

$$\text{for} \quad \left|\overrightarrow{O_1O_2}\right| > \left|\overrightarrow{O_1Q}\right|$$

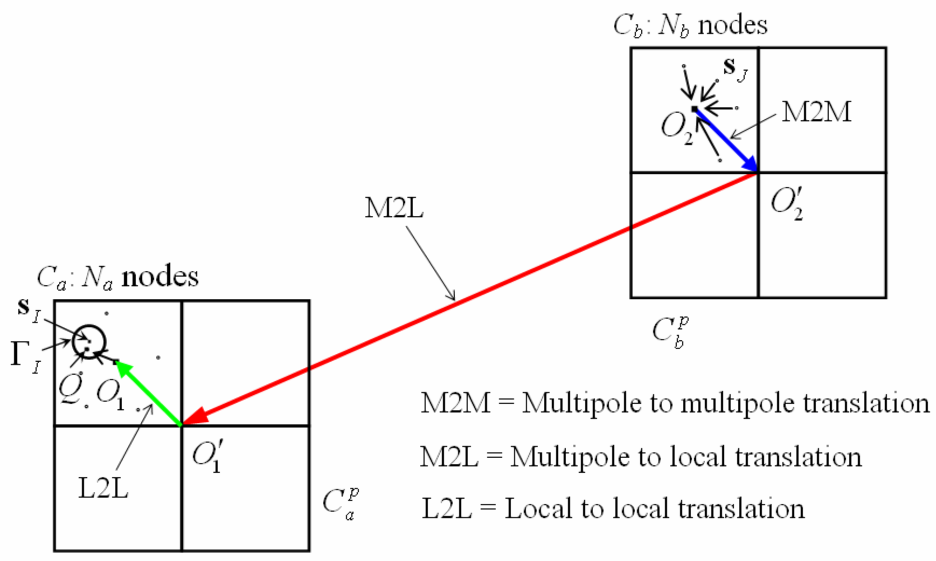
$$\sum_{J=1}^{N_b} \int_{\Gamma_I} \phi_J^s v_I(Q) x_J' d\Gamma = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \int_{\Gamma_I} \frac{1}{4\pi\kappa} R_{n'}^{m'}(\overrightarrow{O_1Q})\, v_I(Q) d\Gamma\, L_{n'}^{m'}(O_1)$$

where

$$L_{n'}^{m'}(O_1) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (-1)^{n'} \overline{S_{n+n'}^{m+m'}(\overrightarrow{O_1O_2})} M_n^m(Q_2)$$

Fast multipole (4)

➢ Translation operators



$$M_{n'}^{m'}(Q_2') = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} R_n^m(\overrightarrow{O_2'O_2})\, M_{n-n'}^{m-m'}(Q_2)$$

Multipole to multipole translation

$$L_n^m(O_1') = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (-1)^n\, \overline{S_{n+n'}^{m+m'}}(\overrightarrow{O_1'O_2'})\, M_{n'}^{m'}(Q_2')$$

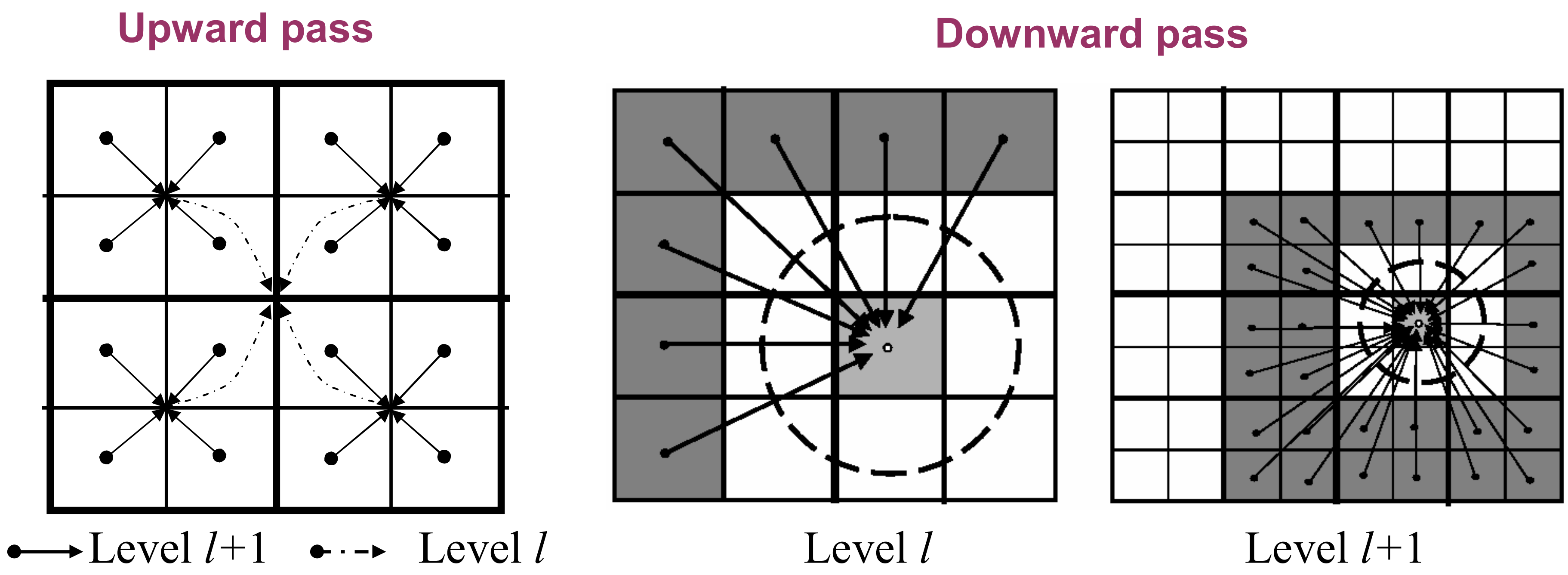
Multipole to local translation

$$L_{n'}^{m'}(O_1) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} R_{n-n'}^{m-m'}(\overrightarrow{O_1'O_1})\, L_n^m(Q_1')$$

Local to local translation

Fast multipole (5)

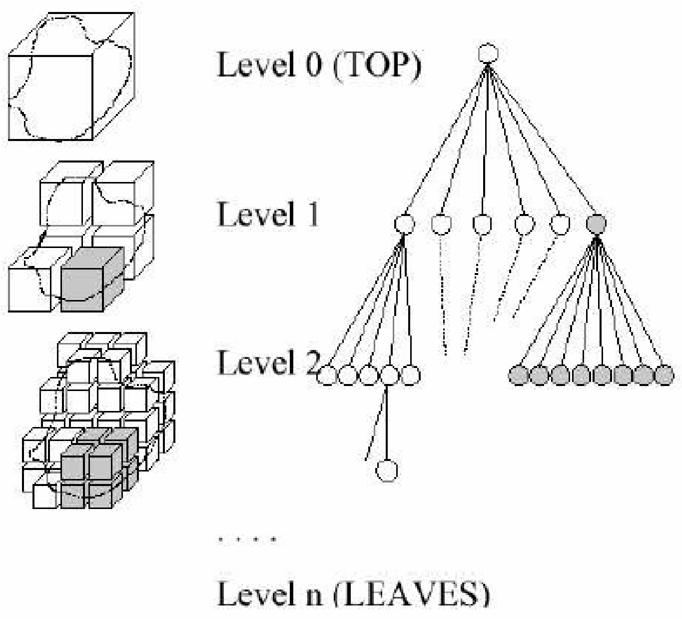
➢ **Recursive algorithm**



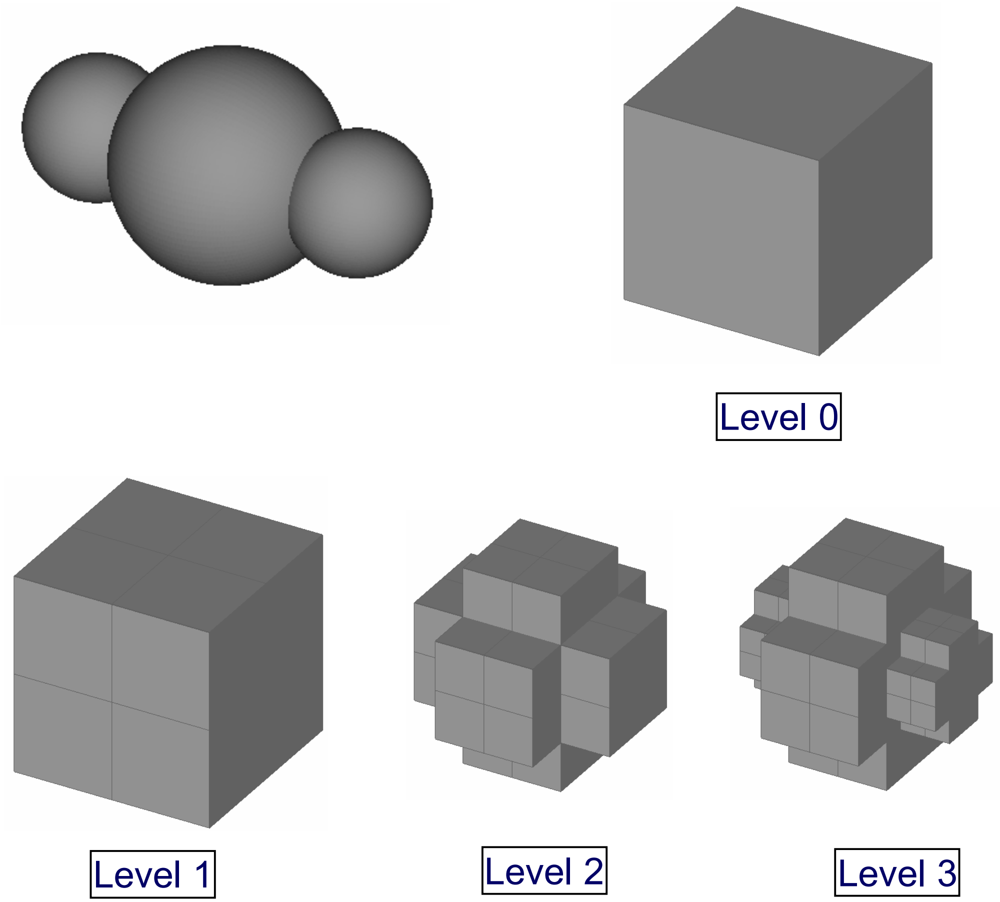
Multipole moments are accumulated from leaves to the root (***Upward pass***); and local moments are distributed from the root to the leaves (***Downward pass***). This is accomplished at a linear complexity.

Standard oct-tree

- **Algorithm**



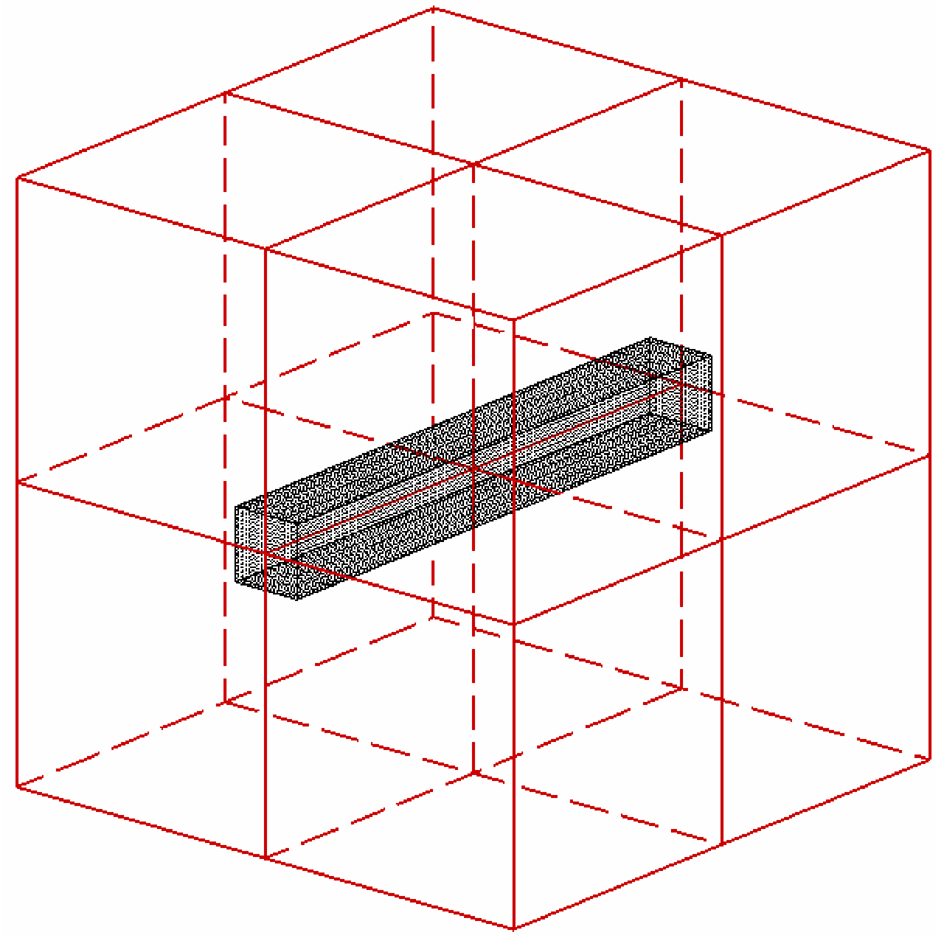
- **One example**



Standard oct-tree (2)

➢ **Shortcoming**

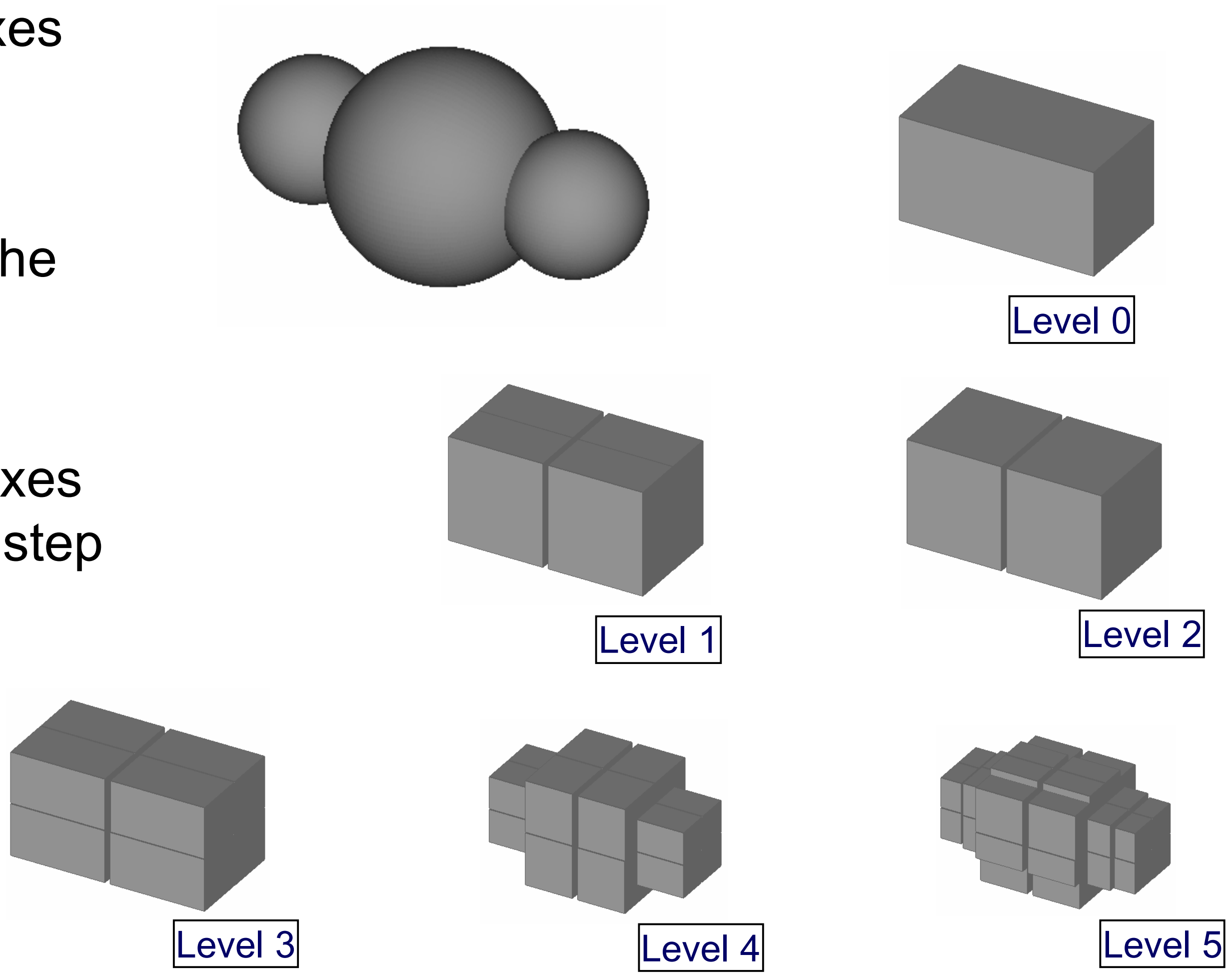
- **Does not reflect the geometry of the computational domain!**
- **Resulted in a large number of M2L translations!**



Binary tree

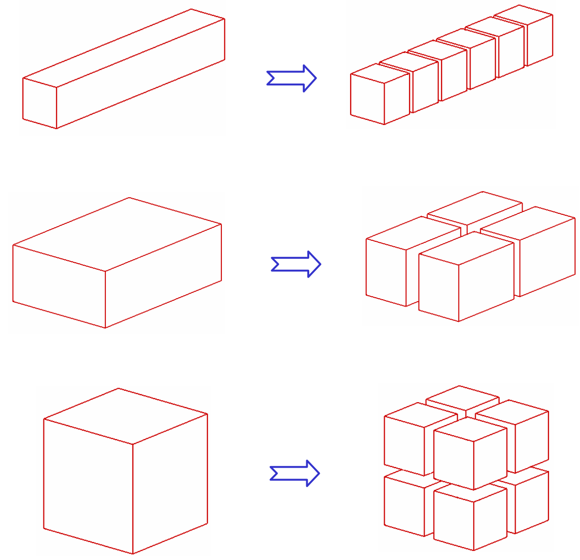
- **Differs from the standard tree:**
  - Use rectangular boxes instead of cubes
  - Subdivide a box in the longest direction
  - Tighten the child boxes at each subdivision step

- **One example**

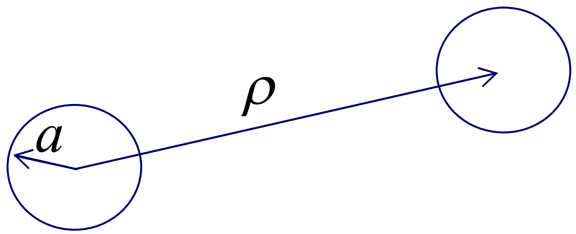


Adaptive tree

- **Differs from the standard tree:**
  - Use rectangular boxes instead of cubes
  - Subdivide a box according to the shape of the box
  - Tighten the child boxes at each subdivision step
  - Generalize the Downward Pass algorithm to allow M2L among the child boxes of a same parent box
  - Determine the number of expansion terms in M2L by

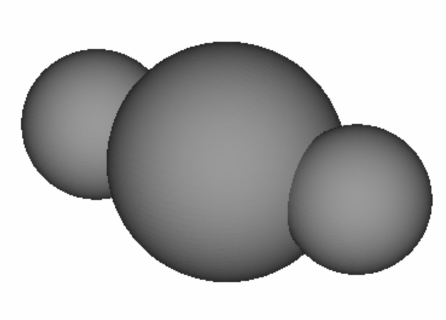


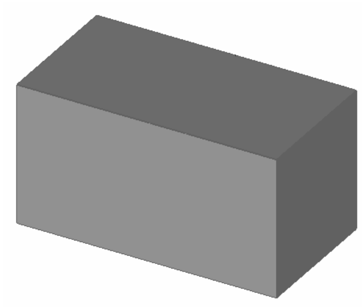
$$p = 0.117\, p_{norm} \Big/ \log\left(\frac{a}{\rho - a}\right)$$



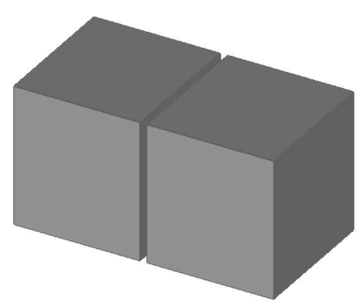
Adaptive tree (2)

➢ **One example**

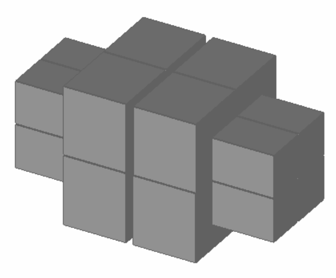




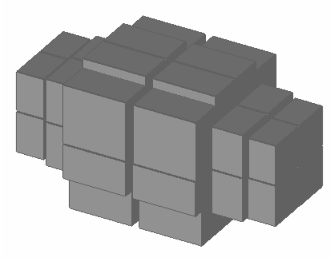
Level 0



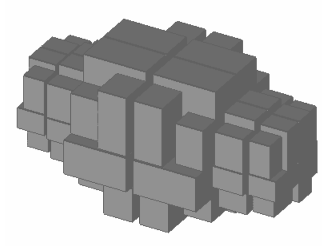
Level 1



Level 2



Level 3



Level 4

Test problems

Computer: desktop computer with an Intel(R) Pentium(R) 4 CPU (1.99GHz)

Analytical solution: $\phi = x^3 + y^3 + z^3 - 3yx^2 - 3xz^2 - 3zy^2$

Maximum nodes in a leaf: 60

Number of expansion terms: $p = 10$

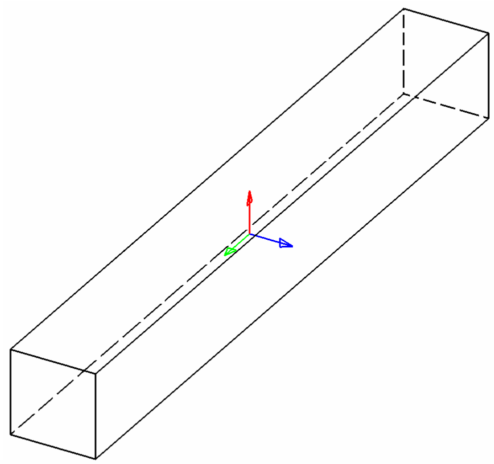
Iterative solver: GMRES

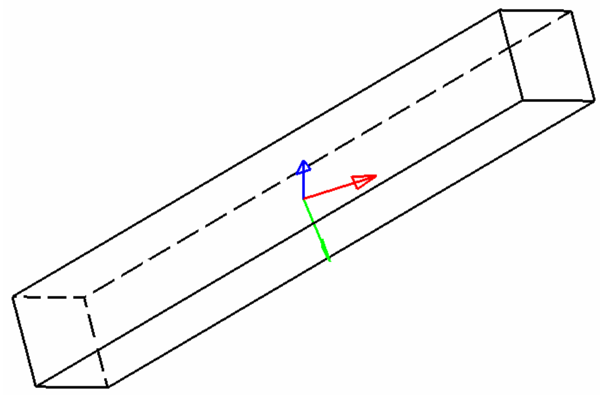
Convergence criterion: relative error $< 10^{-5}$

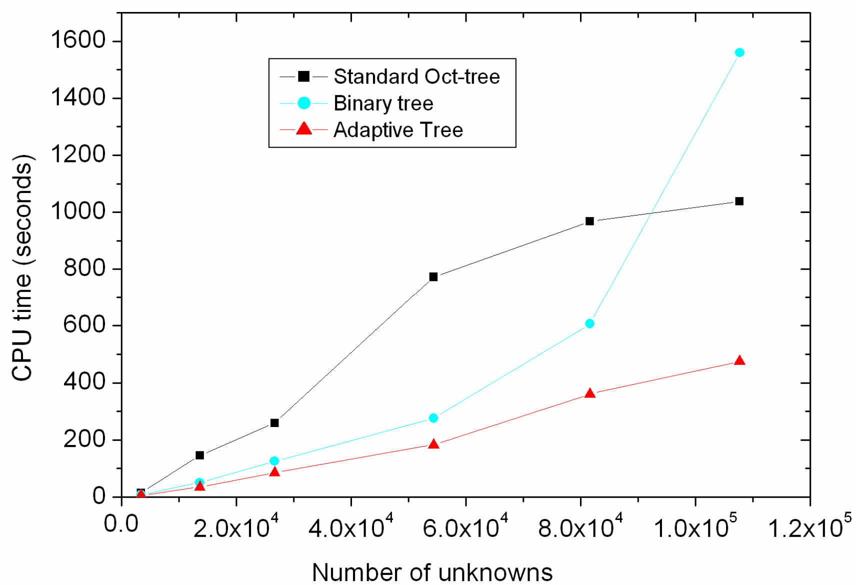
Test problems (2)

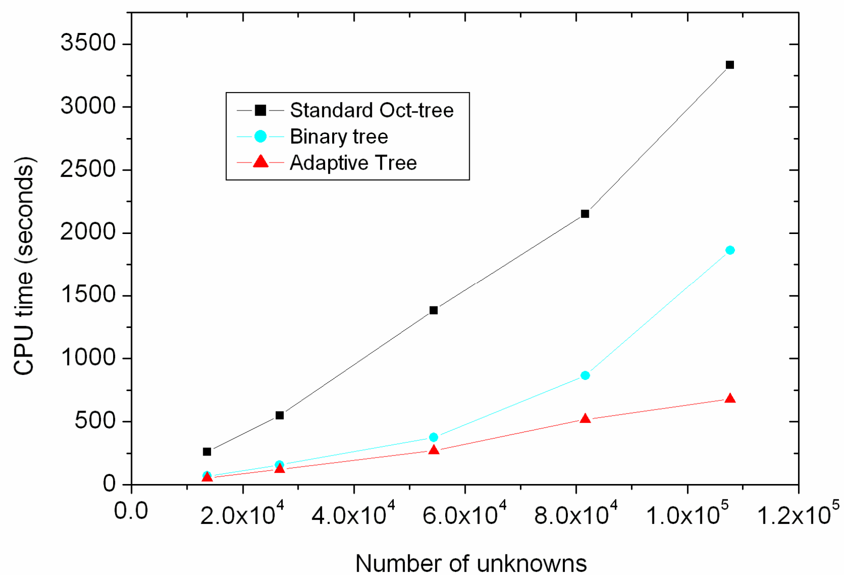
- **A slender box**

Dimensions: 20x20x160

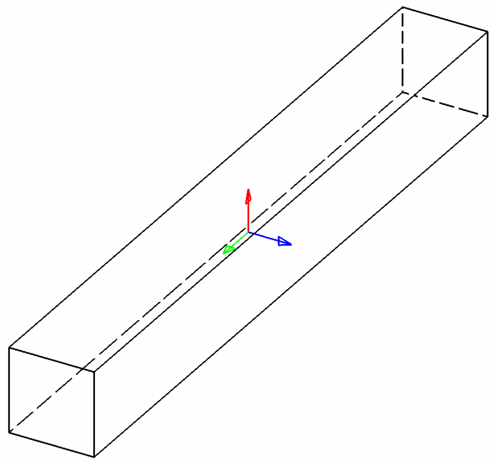


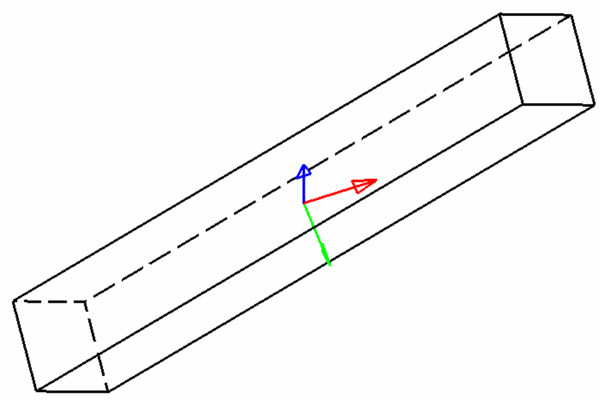


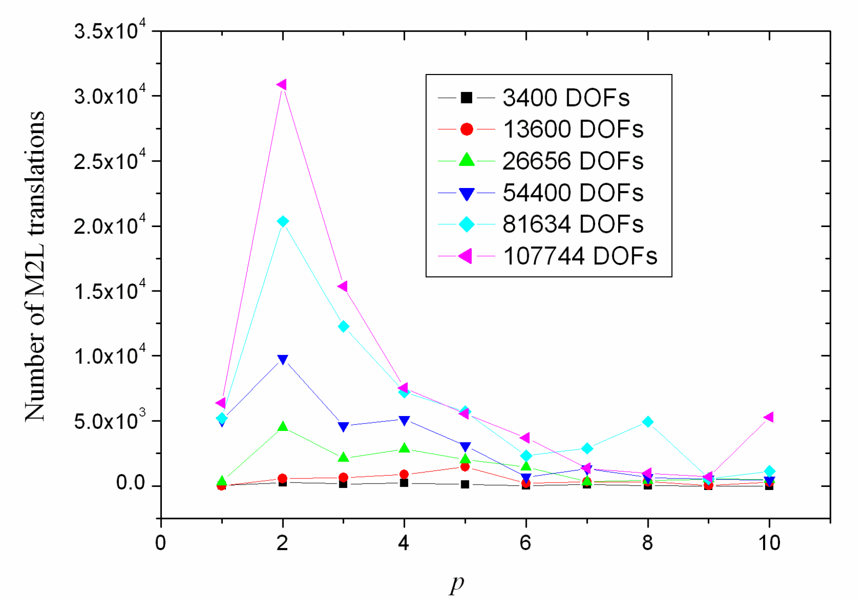


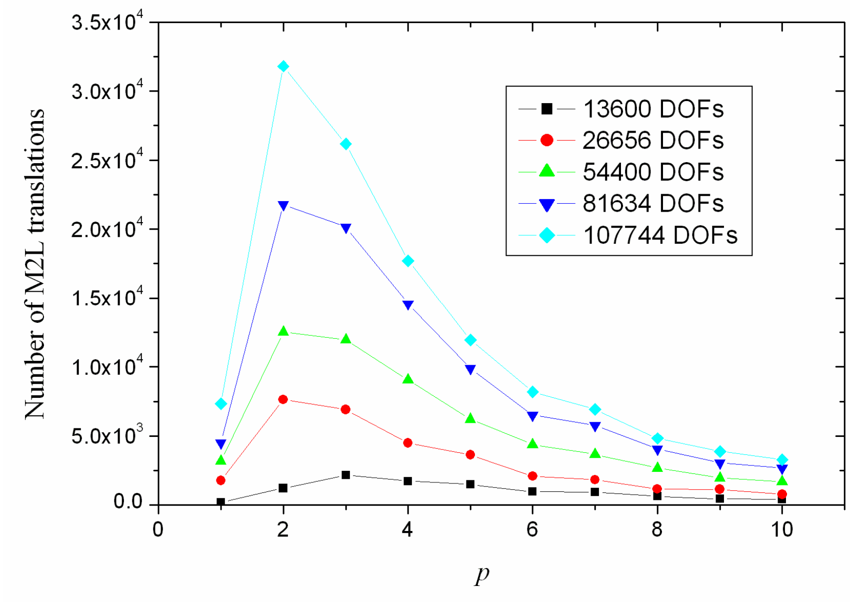


Test problems (3)



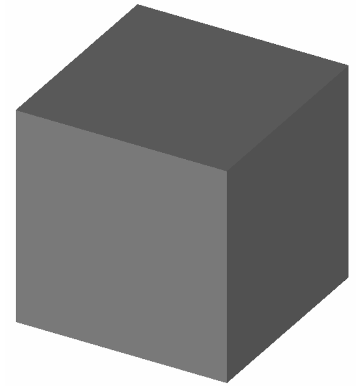


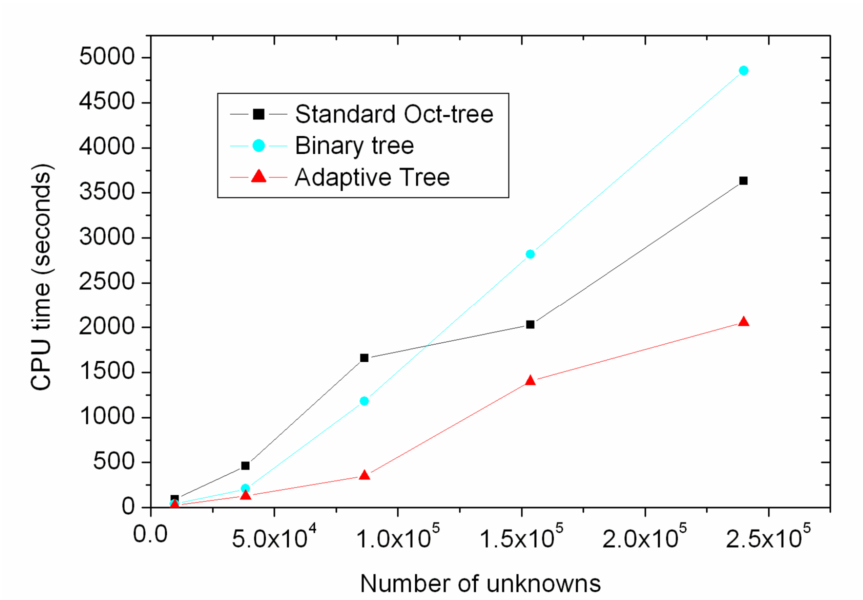




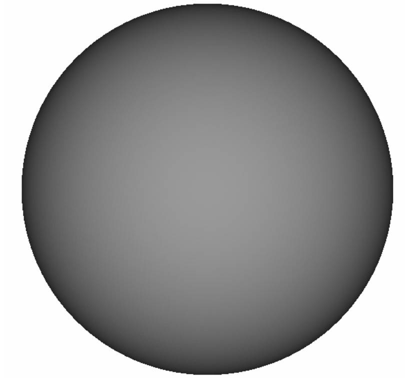
Test problems (4)

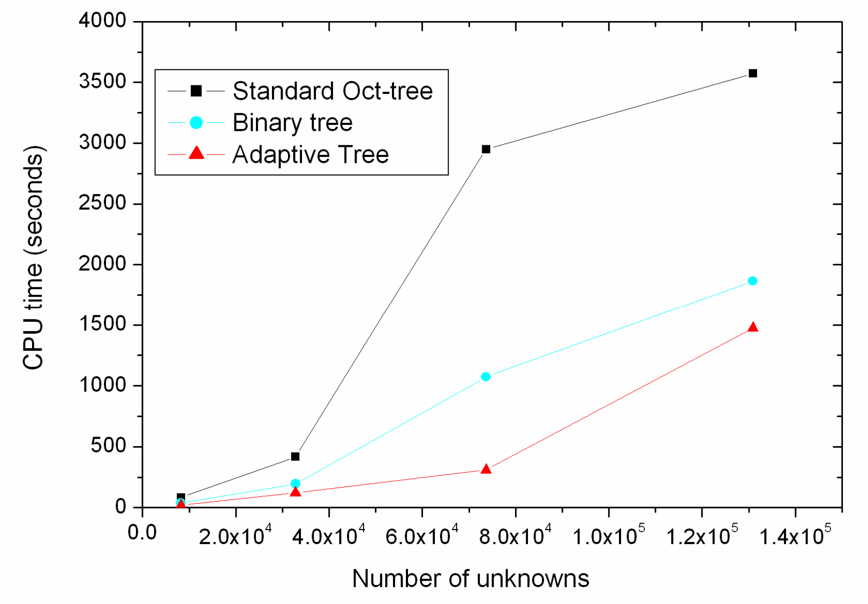
- **A cube**





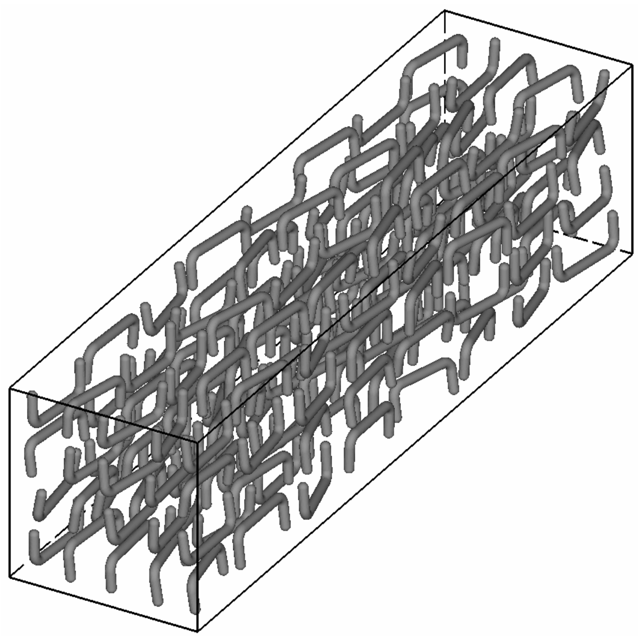
- **A sphere**

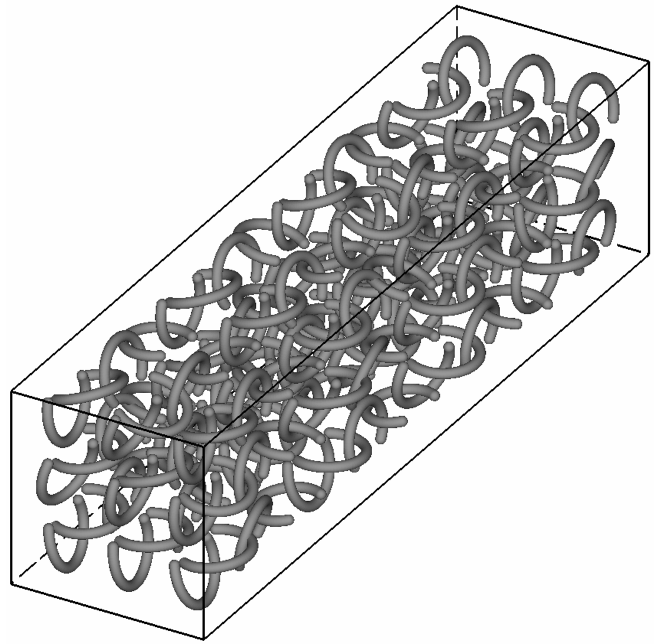




Test problems (5)

➢ **CNT composite simulation**





	κ	Nodes	Time (s)
Oct-tree	1.356	165153	58656
Adaptive tree	1.337	165153	9776

	κ	Nodes	Time (s)
Oct-tree	0.917	109314	32378
Adaptive tree	0.904	109314	5396

Conclusion remarks

- An adaptive tree data structure has been proposed. The new tree data structure is more flexible in matching the geometry (global and local) of the computational domain.

- Moreover, an adaptive value for the number of terms of the truncated series for M2L translations is used. This value is determined by the distance between the two interacting cells.

- Numerical examples show that the adaptive algorithm leads to trees with more compact cells and shallow depth, and runs significantly faster than the standard oct-tree.