Effects of spatial decomposition and *p*-adaptivity on FMM

Jianming Zhang, Masataka Tanaka

Shinshu University
Faculty of Engineering

April 4, 2007

Outline

- Introduction to FMM
- Spatial decomposition in FMM
  - Oct-tree, Binary tree, Adaptive tree
- Definition of cell bounds
- Degree of series expansion
- Strategy for comparative study
- Numerical results
- Conclusions

Introduction-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 direct computation:

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

Fast Multipole Method (FMM):

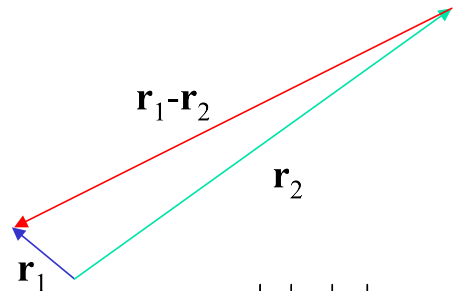
Reducing computational complexity to $O(N)$

Introduction-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

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

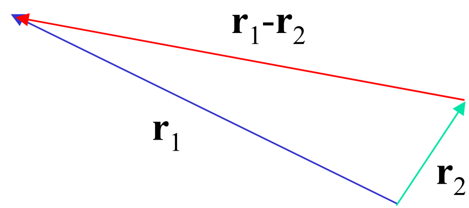


$|\mathbf{r}_1| < |\mathbf{r}_2|$

where

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

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



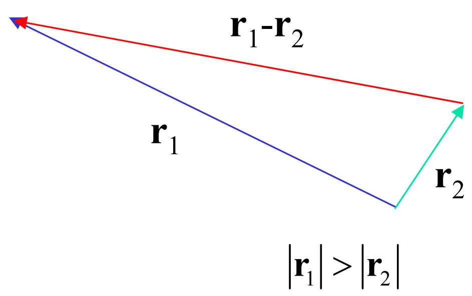
$|\mathbf{r}_2| < |\mathbf{r}_1|$

Introduction-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

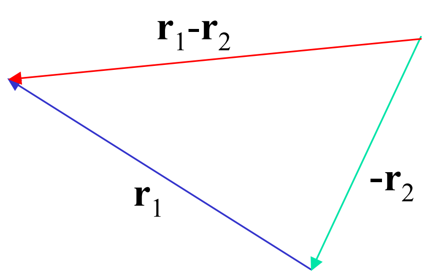
$$S_n^m(\mathbf{r}_1 - \mathbf{r}_2) = \sum_{n'=0}^{\infty} \sum_{m'=-n'}^{n'} \overline{R_{n'}^{m'}(\mathbf{r}_2)} S_{n+n'}^{m+m'}(\mathbf{r}_1)$$



➢Third addition theorem

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

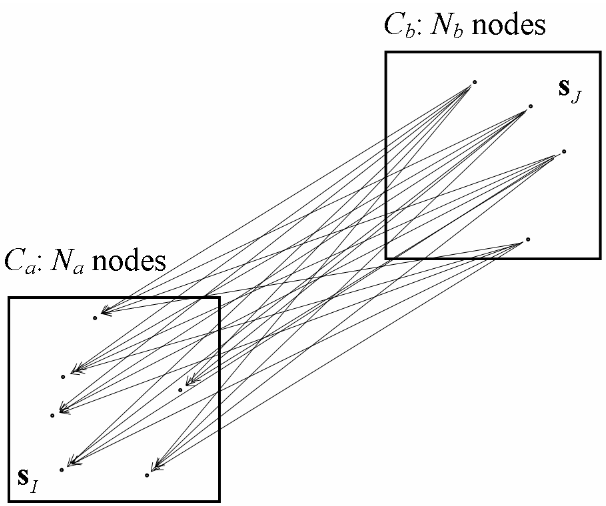
$$R_n^m(\mathbf{r}_1 - \mathbf{r}_2) = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R_{n'}^{m'}(-\mathbf{r}_2) R_{n-n'}^{m-m'}(\mathbf{r}_1)$$



Introduction-cell to cell interaction

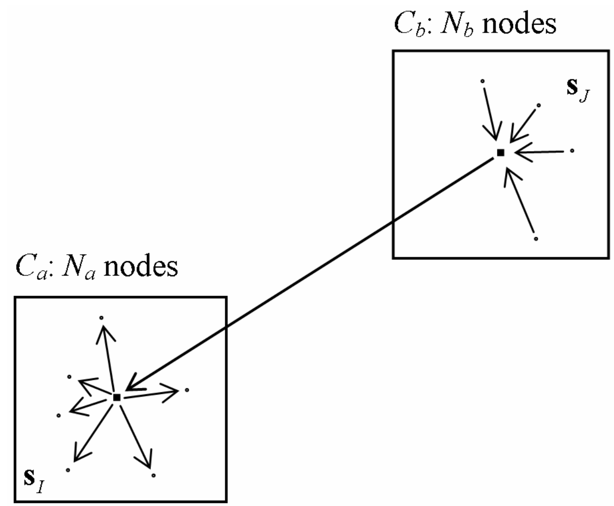
➢ **Ideas of FMM**

Node-node interactions



Complexity $O(N_a N_b)$

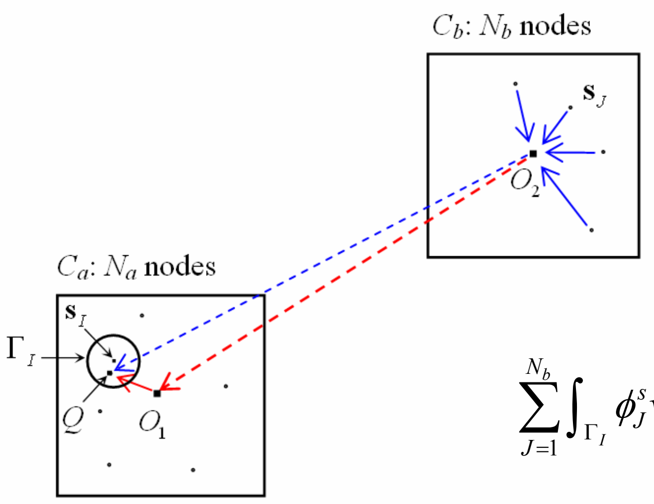
Cell-cell interactions



Complexity $O(N_a+N_b)$

Introduction-Multipole expansion

➢ Multipole expansion



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

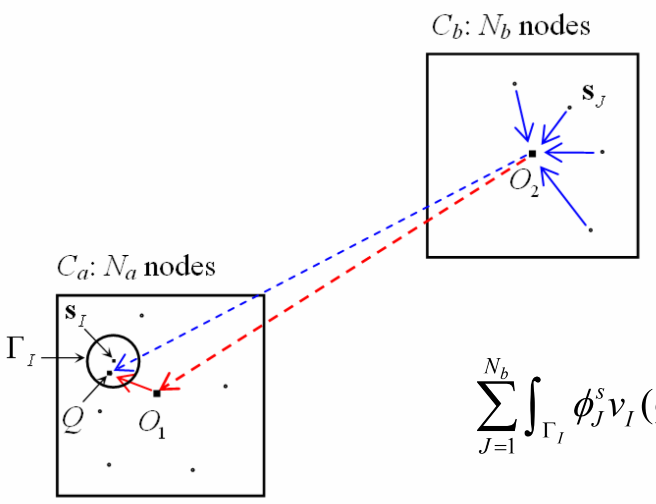
for $\quad \left|\overrightarrow{O_2Q}\right| > \left|\overrightarrow{O_2\mathbf{s}_J}\right|$

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

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

Introduction-Local expansion

- **Local expansion**



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

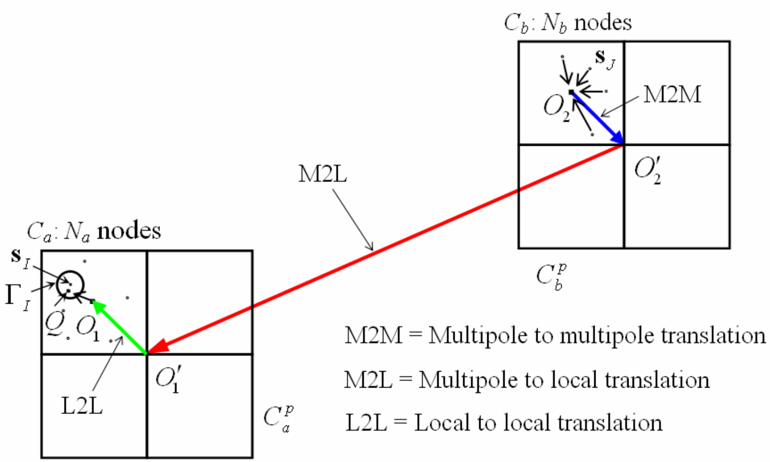
for $\left|\overrightarrow{O_1O_2}\right| > \left|\overrightarrow{O_1Q}\right|$

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

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

Introduction-Translations

➢ 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

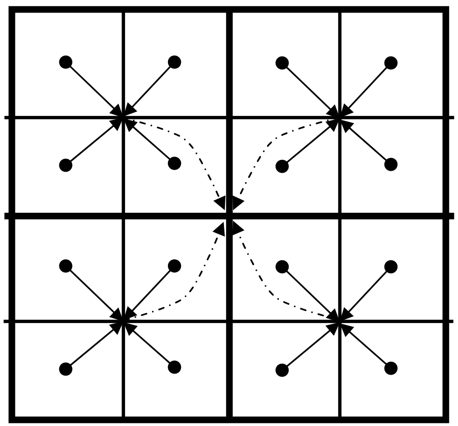
$$L_{n'}^{m'}(O_1) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} R_{n-n'}^{m-m'}(\overrightarrow{O_1'O_1}) L_n^m(Q_1')$$

Local to local translation

Introduction-Fast multipole

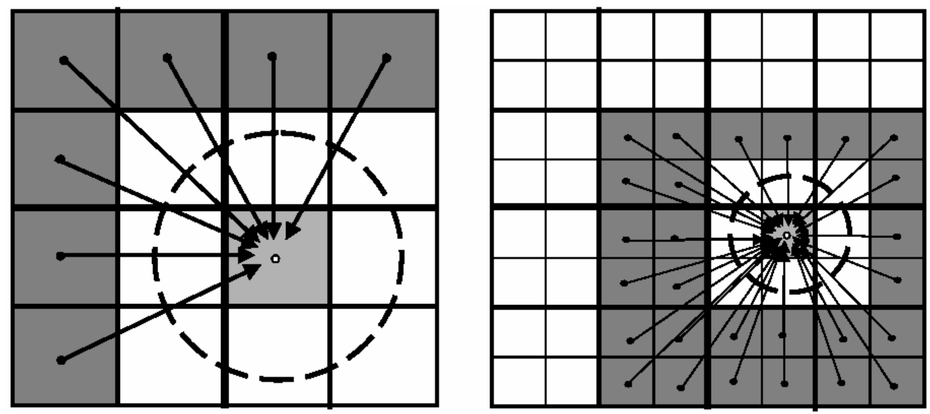
➢ Recursive algorithm

Upward pass



Level l+1 Level l

Downward pass

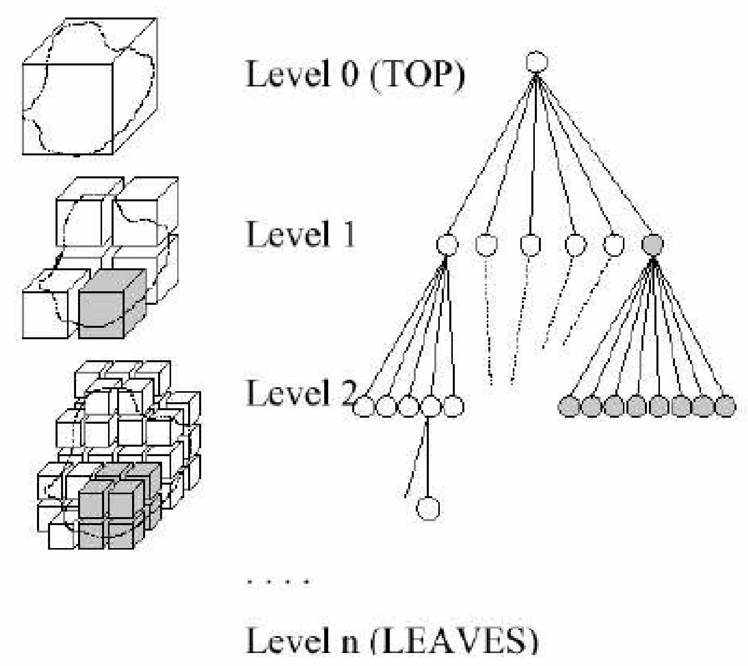


Level l Level l+1

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.

Tree Construction-Oct-tree

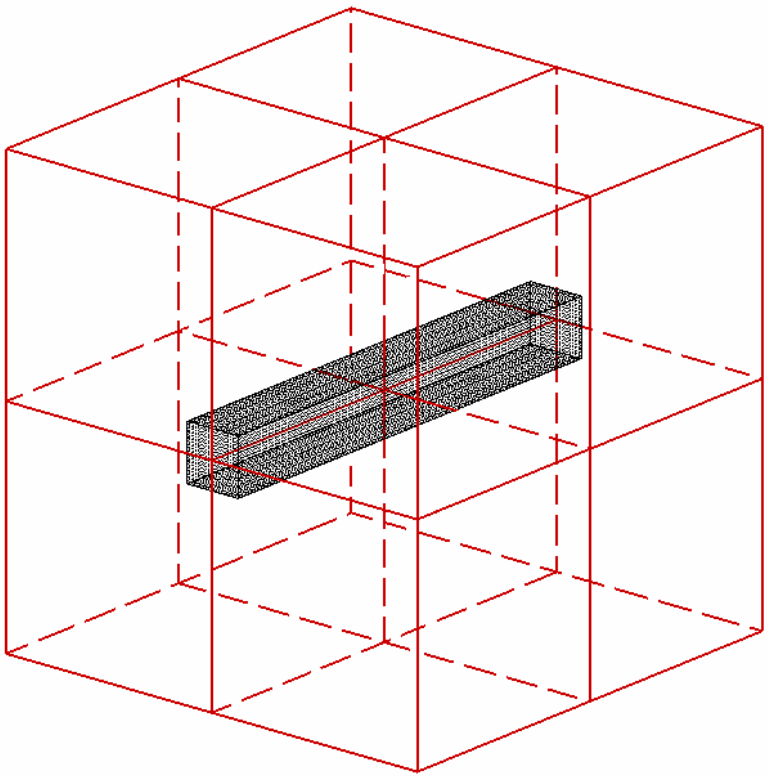
➢ **Algorithm**



Tree Construction-Oct-tree(2)

➢ **Shortcoming**

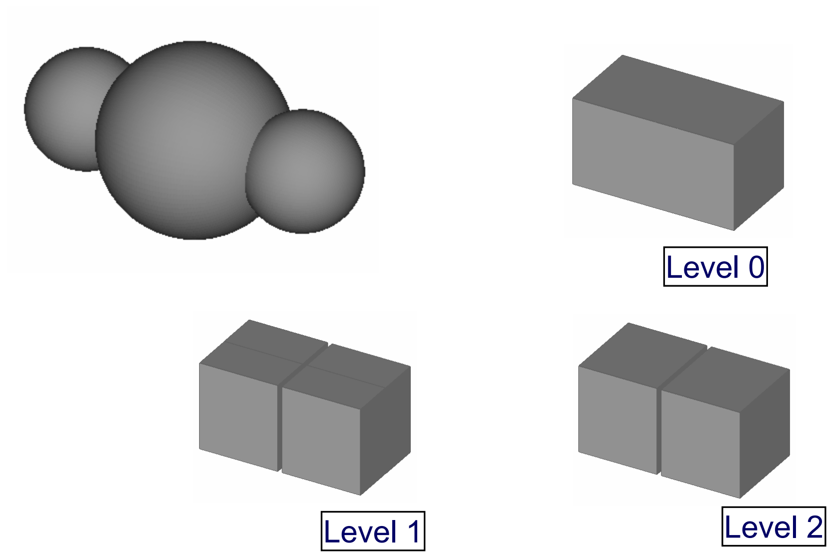
- **Does not reflect the geometry of the computational domain!**
- **Resulted in a large number of M2L translations!**

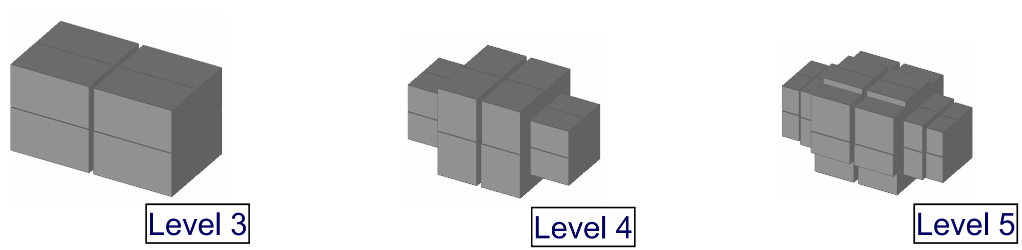


Tree Construction-Binary tree

- **Differs from the oct-tree:**
  - Use rectangular boxes instead of cubes
  - Subdivide a box in the longest direction

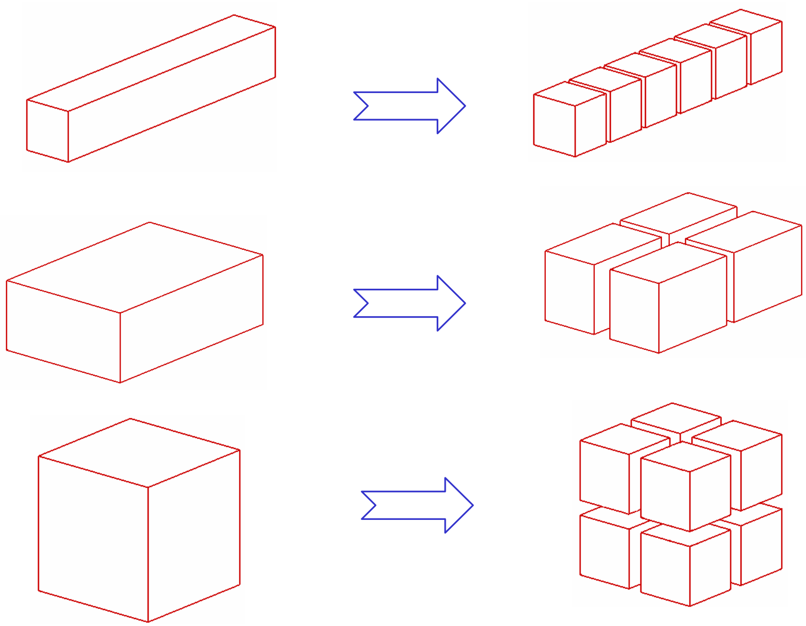
- **One example**





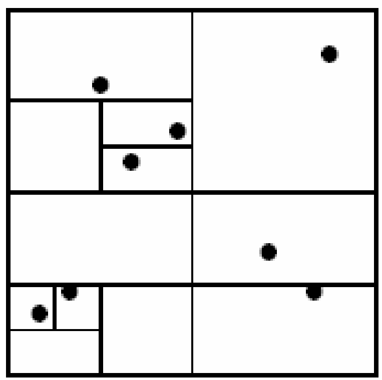
Tree Construction-Adaptive tree

- **Differs from the oct-tree:**
  - Use rectangular boxes instead of cubes
  - Subdivide a box according to the shape of the box

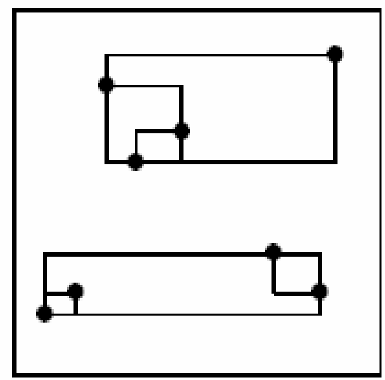


Definition of cell bounds

- **Loose bounds and tight bounds of cells:**



Loose bounds

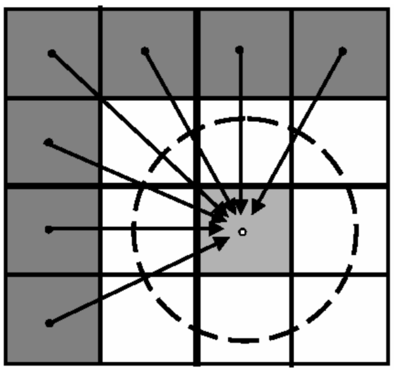


Tight bounds

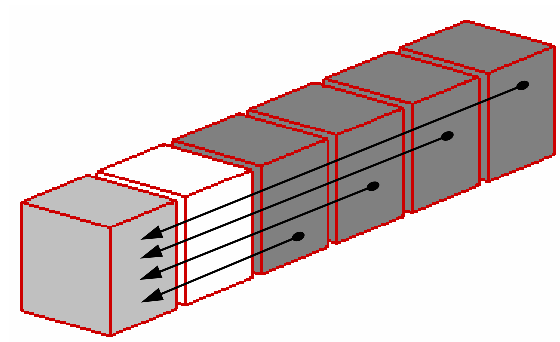
Number of expansion terms

- **Fixed and adaptive number of series expansion terms:**

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

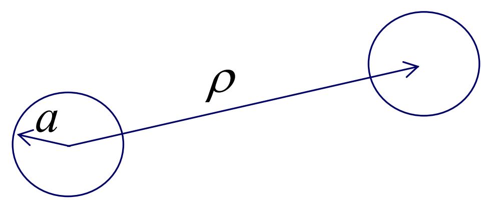


Oct-tree



Adaptive tree

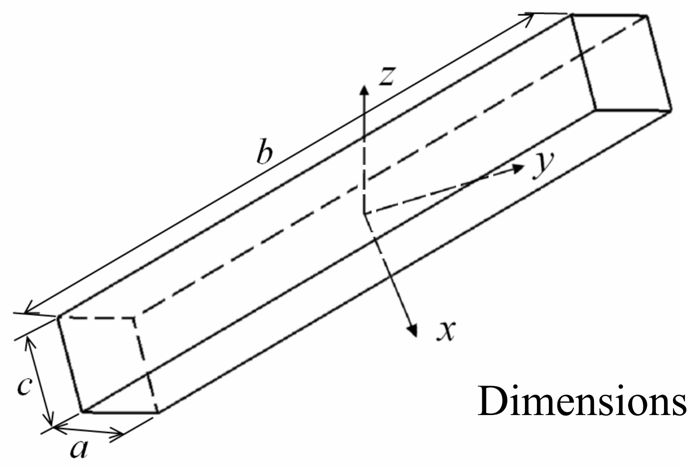
$$p = 0.117\, p_{norm} \Big/ \log\left(\frac{a}{\rho - a}\right)$$

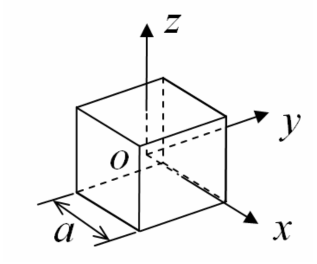


Comparison strategy

- **Three options:**
  - Degree of Tree
    - Oct-tree, Binary tree or Adaptive tree
  - Definition of cell bounds
    - Tight bounds or Loose bounds
  - Number of expansion terms
    - Fixed number or Adaptive number

Test problems





Dimensions: 20x20x160

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

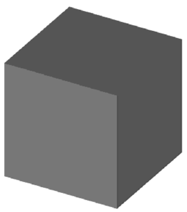
Number of expansion terms: $p = 10$

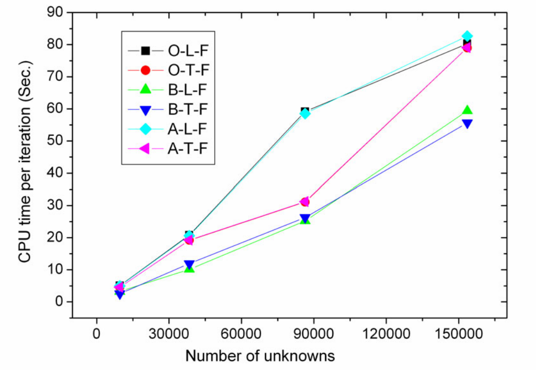
Iterative solver: GMRES

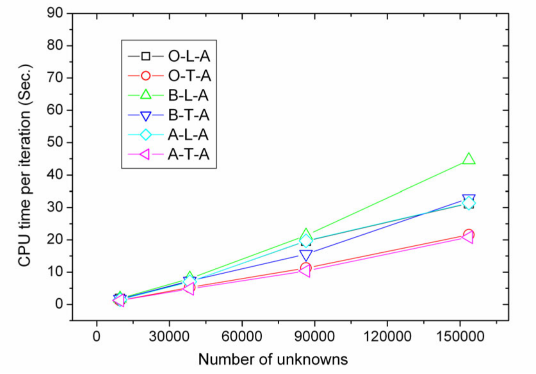
Convergence criterion: relative error $< 10^{-5}$

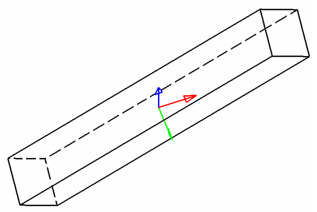
Numerical results

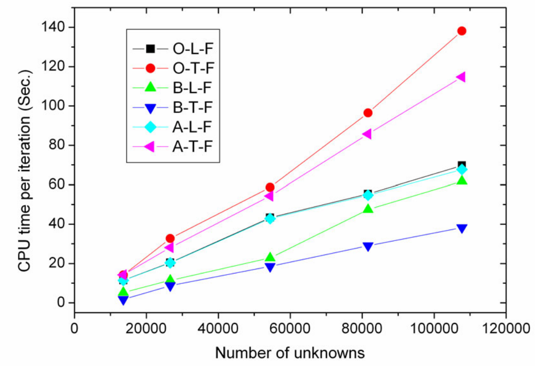
➤ **Number of expansion terms**

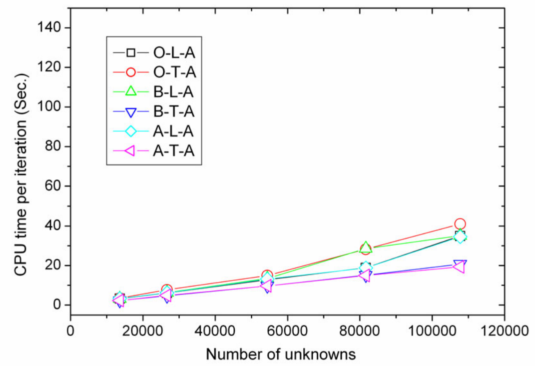






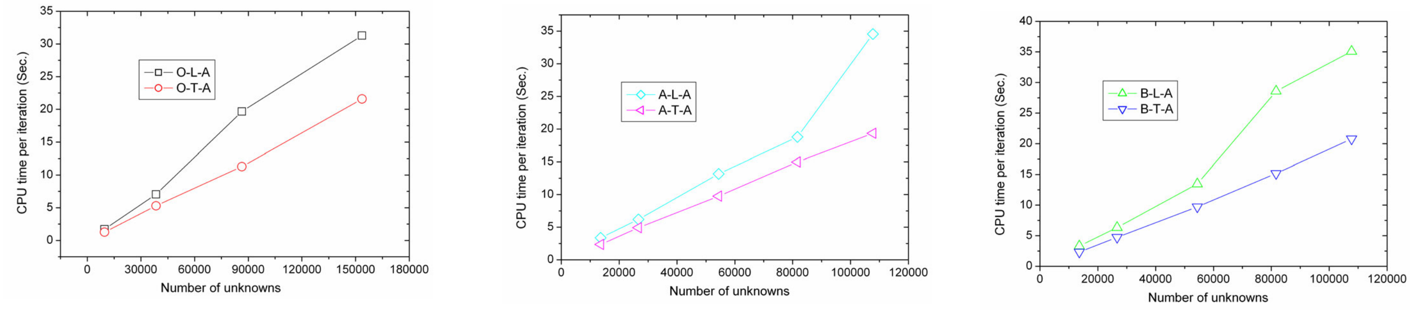




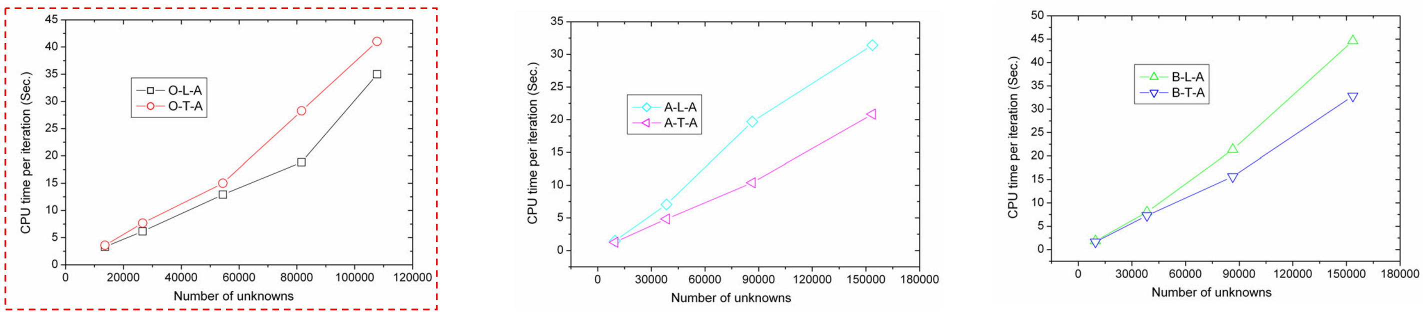


Numerical results (2)

- **Definition of cell bounds**



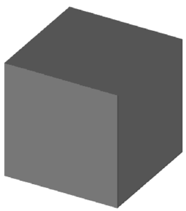
Cube

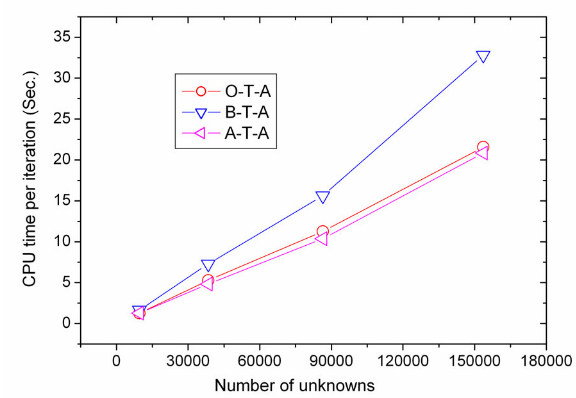


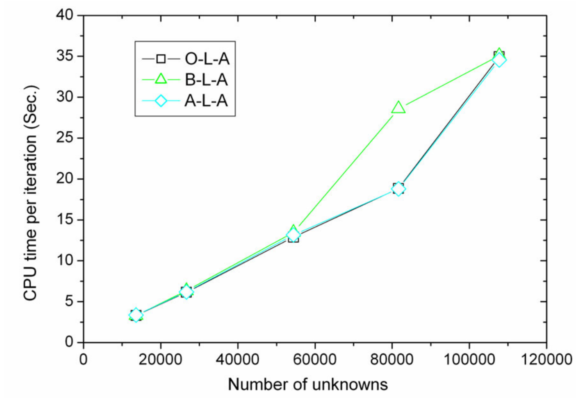
Slender Box

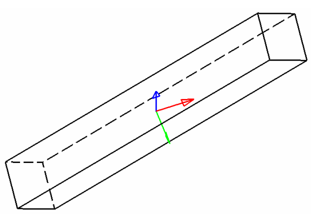
Numerical results (3)

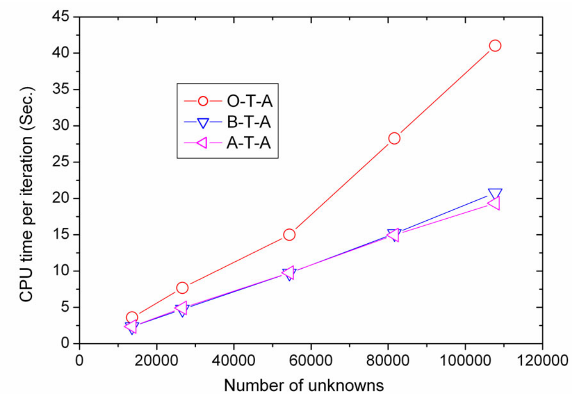
➢ **Degree of Tree**

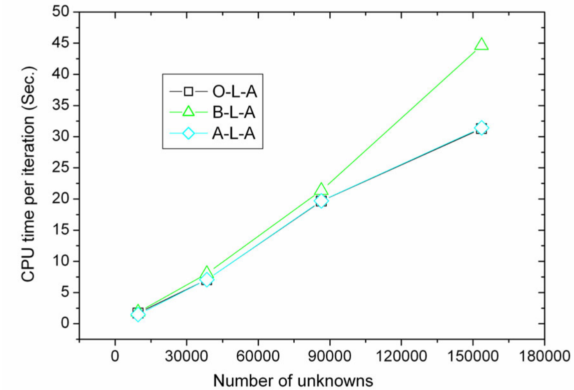






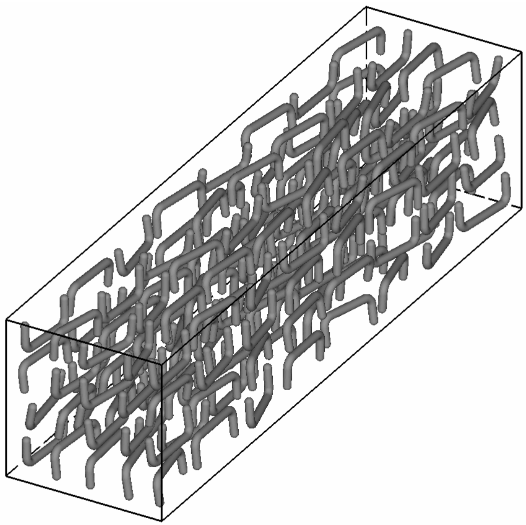


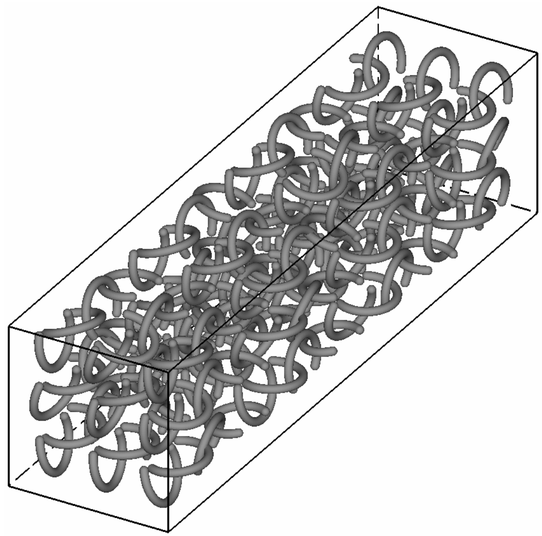




Numerical results (4)

- **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

- We have performed a comparative study on the FMM considering tree degree, cell bounds and series expansion terms.
- The effectiveness of a tree data structure depends on the shape of the computational domain.
- Using adaptive number of expansion terms can always improve computational efficiency.
- The tight cell bounds can be used for binary and the adaptive trees without hurting the performance, while for oct-tree it may cause a substantial slowdown.
- The binary tree is sometimes better and sometimes worse than the oct-tree. The adaptive tree with tight cell bounds and adaptive number of expansion terms is the best algorithm in all cases.