Gridless Fast Multipole Method for Space Charge Effect Simulations

He Zhang

5th ICFA Mini-Workshop on Space Charge 2022

Oct. 24-26, 2022 Knoxville, TN, USA

Introduction of FMM

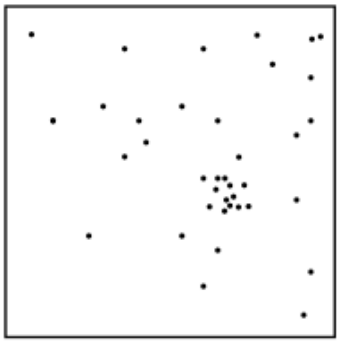
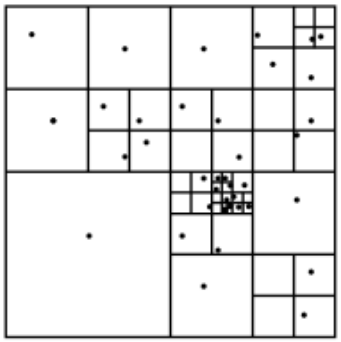
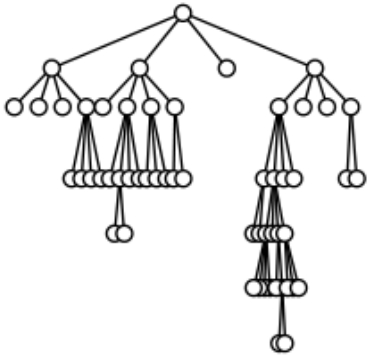
- A fast strategy/framework to calculate the interaction (kernel function) among *N* particles.

- Basic idea of FMM
  - Group the particles by their positions and densities
  - Near region interaction is calculated directly by the pairwise formula
  - Far region interaction is calculated by the multipole/local expansion

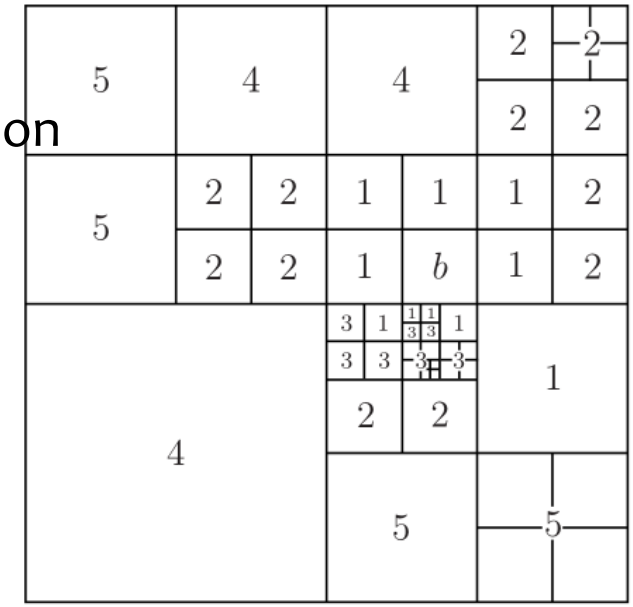
- Property of FMM
  - Scales *O*(*N*) for non-oscillatory kernel with arbitrary particle distribution
  - The error can be strictly estimated and controlled by the order/rank of the expansions.
  - Gridless / tree structure

Multiple Level Fast Multipole Algorithm (MLFMA)

- Group the particles by position and density and represent in a hierarchical partial tree structure.

- Various boxes relations.
- Near region – direction calculation
- Far region – multipole expansion (**M**)/local expansion (**L**)
  - **M** – upwards
  - **L** – downwards
- Hierarchical structure
  - Layer by layer calculation
  - Calculate higher level **M** from the child boxes
  - Distribute higher level **L** to the child boxes



Construct FMM

- There are various ways to construct your expansions
  - Using spherical harmonic functions [1,2]
    - The first FMM
    - Greengard and Roklin
    - https://pypi.python.org/pypi/pyfmmlib
  - Using Taylor expansion
    - Feng Zhao [3]
  - Using Cartesian tensor (Taylor expansion) for r^{-v} with any real v
    - H. Huang and B. Shanker [6]
  - Using TPSA/DA
    - H. Zhang and M. Berz [4,5]

Cartesian Tensor Based FMM: Basic Idea

- Consider the Taylor expansion of a function:

$$f(\mathbf{r}-\mathbf{r}') = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \mathbf{r}'^n : \boldsymbol{\nabla}^n f(\mathbf{r}).$$

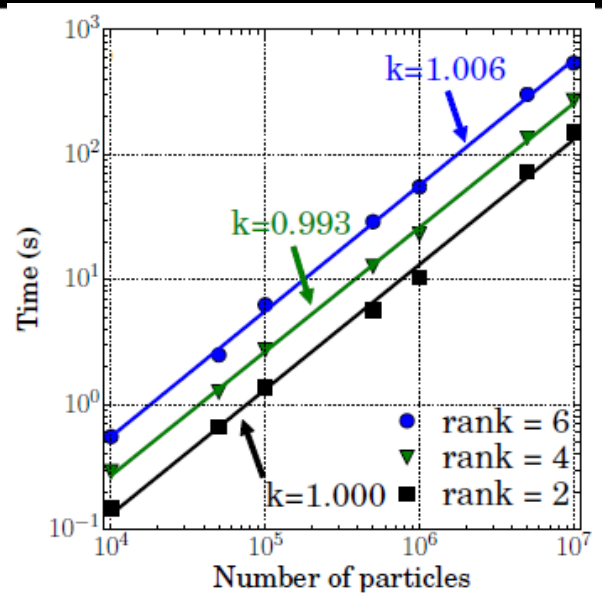
- It can be written in the format of Cartesian Tensor:

$$f(\mathbf{r}-\mathbf{r}') = \begin{cases} \sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \boldsymbol{\nabla}^n f(\mathbf{r}) & \text{for } \mathbf{r} > \mathbf{r}' \\ \sum_{n=0}^{\infty} \mathbf{r}'^n : \mathbf{L}^{(n)} & \text{for } \mathbf{r}' > \mathbf{r} \end{cases}, \qquad \begin{aligned} \mathbf{M}^{(n)} &= \frac{(-1)^n}{n!} \mathbf{r}'^n, \\ \mathbf{L}^{(n)} &= \frac{(-1)^n}{n!} \boldsymbol{\nabla}^n f(\mathbf{r}). \end{aligned}$$

  ":" means contraction of tensors.

- Calculate the high order gradients of $f(\mathbf{r})$.
- Coulomb kernel $f(\mathbf{r}) = 1/r$ – analytical formula exists for the gradients.

Numerical Examples for 1/r FMM



Computation time for different particle numbers and different ranks

N	rank 2 T_ϕ (s)	rank 4 T_ϕ (s)	rank 6 T_ϕ (s)
10,000	0.148	0.290	0.554
50,000	0.656	1.259	2.494
100,000	1.367	2.728	6.283
500,000	5.703	12.790	29.092
1,000,000	10.326	23.178	55.206
5,000,000	73.300	136.147	304.463
10,000,000	151.747	270.753	541.240

Relative errors for different ranks

rank	2	4	6	8	10
σ_ϕ	1.825×10^{-3}	1.345×10^{-4}	2.523×10^{-5}	6.517×10^{-6}	2.720×10^{-6}

Potential calculation for 1,000,000 particles of normal distribution

	pyfmmlib		Traceless Cartesian tensor FMM			
iprec	σ_ϕ	T_ϕ (s)	rank	s	σ_ϕ	T_ϕ (s)
-2	7.78×10^{-4}	29.70	3	64	4.09×10^{-4}	18.71
-1	1.00×10^{-4}	37.11	4	64	1.10×10^{-4}	26.74
0	2.03×10^{-6}	60.88	10	256	2.27×10^{-6}	159.25

- I7-3630QM CPU at 2.4 GHz

Boundary Value Problem

- Combine FMM with boundary element method (BEM)
- Boundary Integral Equation (BIE):

$$c(\mathbf{r})\phi(\mathbf{r}) = \int_S \left[G(\mathbf{r}, \mathbf{r}')q(\mathbf{r}') - F(\mathbf{r}, \mathbf{r}')\phi(\mathbf{r}')\right] \mathrm{d}S(\mathbf{r}') + \int_V G(\mathbf{r}, \mathbf{r}')f(\mathbf{r}')\mathrm{d}V(\mathbf{r}'), \quad \mathbf{r} \in S,$$

- Discrete the BIE on the boundary, we get

where $c(\mathbf{r}) = 1/2$ if S is smooth around $\mathbf{r}$. V is the domain and S is the boundary

$$\begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \cdots & f_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N1} & f_{N2} & \cdots & f_{NN} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N1} & g_{N2} & \cdots & g_{NN} \end{pmatrix} \cdot \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{pmatrix} + \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{pmatrix},$$

where

$$g_{ij} = \int_{\Delta S_j} G(\mathbf{r}_i, \mathbf{r}')f(\mathbf{r}')\mathrm{d}S, \quad f_{ij} = \frac{1}{2}\delta_{ij} + \int_{\Delta S_j} F(\mathbf{r}_i, \mathbf{r}')f(\mathbf{r}')\mathrm{d}S, \quad s_i = \int_V G(\mathbf{r}_i, \mathbf{r}')f(\mathbf{r}')\mathrm{d}V(\mathbf{r}'),$$

Boundary Value Problem

- Reorganize the above equations, we get the following linear system, which can be solved iteratively.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}, \text{ or } \mathbf{A} \cdot \boldsymbol{\lambda} = \mathbf{b},$$

where $\boldsymbol{\lambda}$ is the unknown vector and $\mathbf{b}$ is the known vector.

- The linear system can be solved iteratively.
- FMM is used to accelerate the calculation of the matrix.
- Note: Even for electrostatic BVP, Coulomb kernel is not enough for BEM. FMM for general kernels needs to be used.

FMM for Arbitrary Kernel

- FMM is not limited for Coulomb interaction or gravity (1/r format).
- Any non-oscillatory kernel function can be calculated by FMM
- We need to figure out a way to construct the expansions for a kernel that you do not know beforehand.
- There are various ways :
  - A general FMM, by Z. Gimbutas and V. Rokhlin [8]
  - Kernel-Independent FMM, by L. Ying [9]
  - Black-box FMM, by E. Darve [11]
  - Cauchy Integral FMM, by E. Darve [12]
  - Cartesian tensor and TPSA/DA by H. Zhang & H. Huang
  - Python code generator

Cartesian Tensor based FMM for Arbitrary Kernel

- Taylor expansion for a function $f(\mathbf{r})$:

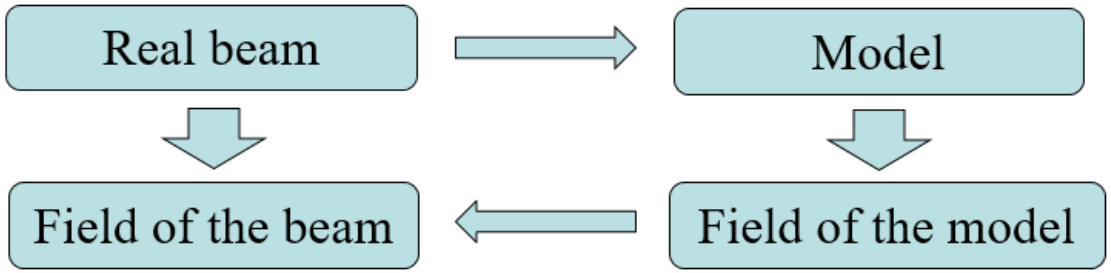
$$f(\mathbf{r}-\mathbf{r}') = \begin{cases} \sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \boldsymbol{\nabla}^n f(\mathbf{r}) & \text{for } \mathbf{r} > \mathbf{r}' \\ \sum_{n=0}^{\infty} \mathbf{r}^n : \mathbf{L}^{(n)} & \text{for } \mathbf{r}' > \mathbf{r} \end{cases},$$

- By nature it works for any function as long as we can calculated $\nabla^n f(\mathbf{r})$.
- Whenever we need to calculate $\nabla^n f(\mathbf{r})$, the value of $\mathbf{r}$ is known.
- Truncated Power Series Algebra (TPSA) or Differential Algebra (DA)

Parallelization of FMM

- Parallelization is challenging due to the unbalanced partial tree.
- But FMM parallelizes well on thousands or even more of nodes
- Some libraries:
  - PVFMM: Parallel kernel-independent fmm via MPI (distributed structure)
    - https://github.com/dmalhotra/pvfmm
    - Blood flow simulation, 90 billion unknowns simulated in each step time, won 2010 Gordon Bell Prize, parallel on 200,000 AMD notes on ORNL Jaguar PF system [10] (pkifmm)
  - Exafmm: Excellent parallel scaling via MPI (distributed structure)
    - Supporting various kernels
    - https://github.com/exafmm/exafmm
    - ExaFMM-t, C++&Python, OpenMP for shared memory, 0.95/1.48 s for 7/10 digits accuracy for 1M particles using 14-core i9-7940x, 2022
- Examples of recent works:
  - 2018 in biomolecular hydrodynamic simulation, 15 M particle, 12288 cores, 54% strong-scaling efficiency [21]
  - 2017 in AI kernel evaluation, 11 M x 11 M kernel matrix in 2 mins on 3072 x86 Haswell cores, 4.5 M x 4.5 M matrix in 1 mins on 4352 “Knights Landing” cores. [22]

Is FMM Noisy ?



- There are two steps :1. Make the model of the beam; 2. Calculate the field of the model.
- Point charges – fluctuation of field is expected when charges are very close. FMM calculates the field of your model **honestly**.
- To obtain smooth field - use small spherical Gaussian bunch instead of point charges or set a cut off distance for your point charge potential/field calculation.
- Our responsibility to choose a proper model of the beam.

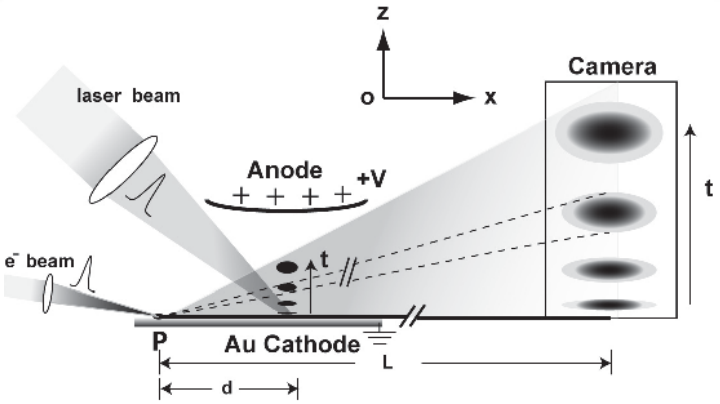
Implementation in Accelerator Physics

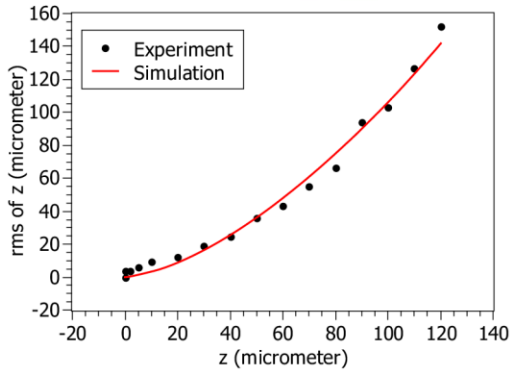
- Gridless, O(n) for arbitrary distribution - complex charge distribution or boundary
- Accurate near field calculation – micro-dynamics, collisional process
- Good benchmark for PIC code

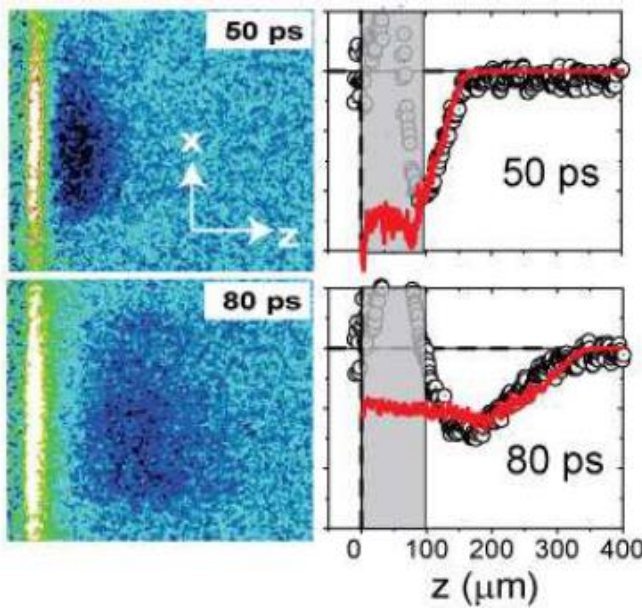
- Some examples:
  - Space charge dominated photoemssion process simulation - MSU, 2012
  - Space charge effect in FAIR SIS100 – GSI, ICAP'18
  - Space charge and electron cooling – NIU
  - MATCH-B, FMM in Synergia – Reservoir Labs Inc, IPAC'21

- Open questions:
  - Not many ready-to-use tools available
  - Numerical performance/properties not fully understood
  - Not many users

Use FMM in Accelerator Study: An Example

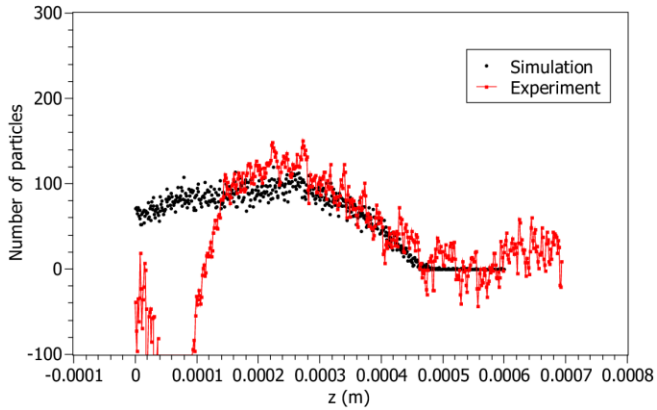


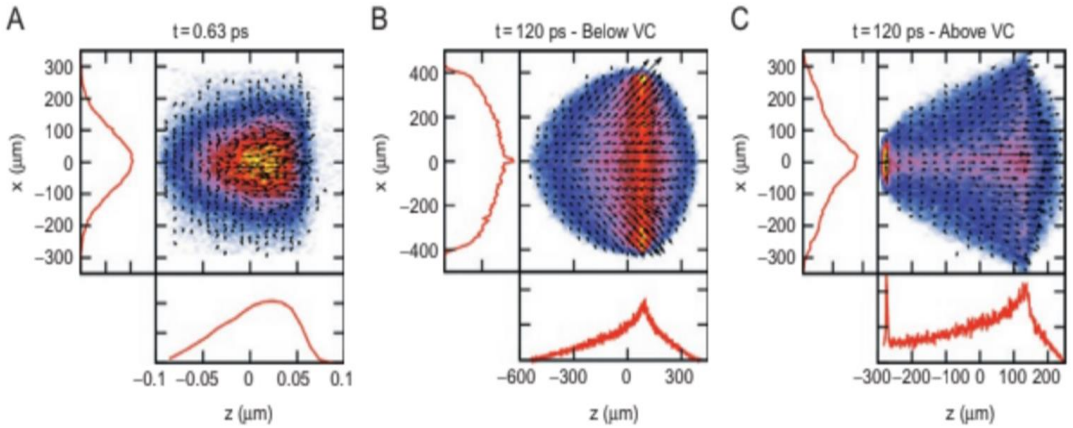




Simulation of the Photoemission process:

- 2,000,000 3d Gaussian macro-particle.
- 8th order Runge-Kutta integrator





Space charge simulations of photoemission using the differential algebra-based multiple-level fast multipole method, H. Zhang, Z. Tao, C.-Y. Ruan, et. al., *Microscopy and Microanalysis*, 21, 224., 2015

Multiscale modeling of the ultrafast electron microscope: from the photocathode to the sample, J. Portman, H. Zhang, K. Makino, et. al., *Advances in Imaging and Electron Physics* 191 (2015) 117-130.

Summary slide, 5th ICFA mini-workshop on Space Charge
Theme: Bridging the gap in space charge dynamics

In 1-2 sentences, summarize the content of this presentation

- Introduce the FMM method with numerical examples.
- Good tool for complex charge distribution and boundary
- Useful when micro-dynamics between particles is desired, especially for collisional process (beyond space charge)

From your perspective, where is the gap regarding space charge effects? (understanding/control/mitigation/prediction/?)

- Enhance the simulation capability for complex cases.

What is needed to bridge this gap?

- Innovative algorithm and high efficiency parallel codes


谢
谢
THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
BÏYAN
SHUKRIA
TINGKI
YAQHANYELAY
TASHAKKUR ATU
SUKSAMA
EKHMET
GRAZIE
MEHRBANI
MAAKE
KOMAPSUMNIDA
JUSPAXAR
GOZAIMASHITA
EFCHARISTO
PALDIES
BOLZÏN
MERCI
SPASSIBO
SHACHALHUYA
NURUN
CHALTU
WABEEJA
MAITEKA
YUSPAGARÀTAM
HUI
DHANYABAAD
ANIHA
ATTO
SPASIBO
DENKUJA
NENACHALHYA
UNALCHÉESH
GUI
HATUR
EKOJU
SIKOMO
MERASTAHHY
SAHÇO
GAEJTHO
LAH
TAVYAPUCH
MEDAWAGSE
AGUYJE
FAKAAUE
MAKETAI
MINMONCHAR

References

1. A fast adaptive multipole algorithm for particle simulations, J Carrier, L Greengard, V Rokhlin, SIAM journal on scientific and statistical computing 9 (4), 1988
2. A fast Adaptive Multipole Algorithm in Three Dimensions, H. Cheng, L. Greengard, and V. Rokhlin, Journal of computational physics 155.2, 1999
3. An O(N) algorithm for three-dimensional n-body simulations, Feng Zhao, MIT master thesis, 1987
4. The fast multipole method in the differential algebra framework, H. Zhang, M. Berz, NIM A, 2011
5. Space charge simulations of photoemission using the differential algebra-based multiple-level fast multipole method, H. Zhang, Z. Tao, C.-Y. Ruan, et. al., Microscopy and Microanalysis, 21, 224., 2015
6. Accelerated Cartesian expansion - A fast method for computing of potentials of the form R^-v for all real v, B. Shanker, H. Huang, JCOMP, 2007.
7. A novel wideband FMM for fast integral equation solution of multiscale problems in Electromagnetics, M. Vikram, H. Huang, B. Shanker, T. Van, Transactions on antennas and propagation, 2009
8. A generalized fast multipole method for non-oscillatory kernels, Z. Gimbutas and V. Rokhlin, SIAM Journal on Scientific Computing 24.3, 2000

References

9. A kernel-independent adaptive fast multipole algorithm in two and three dimensions, Lexing Ying, George Biros, Denis Zorin, JCOMP, 2004
10. Petascale direct numerical simulation of blood flow on 200K cores and heterogeneous architectures, A. Rahimian, I Lashuk, S Veerapaneni, et al. Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society, 2010
11. The black-box fast multipole method, William Fong, Eric Darve, JCOMP, 2009
12. Fast multipole method using the Cauchy integral formula, C Cecka, P.-D. Letourneau, E. Darve, Numerical Analysis of Multiscale Computations (pp. 127-144). Springer, Berlin, Heidelberg., 2012
13. 42 TFlops hierarchical n-body simulations on GPUs with applications in both astrophysics and turbulence, T. Hamada, T. Narumi, R. Yokota, et al. High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on. IEEE, 2009.
14. Fast multipole accelerated boundary integral equation methods, N Nishimura, Appl. Mech. Rev 55(4), 299-324, 2002
15. Diagonal forms of translation operators for the Helmholtz equation in three dimensions, V. Rokhlin, Appl. Comput. Harmon. Anal. 1 (1) (1993) 82-93
16. Fast directional algorithms for the Helmholtz kernel, B. Engquist, L. Ying, Journal of computational and applied mathematics, 2010

References

17. Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation, M. Messner, M. Schanz, E. Darve, JCOMP, 2012
18. Fast multipole method solution of three dimensional integral equation, J.M. Song, W.C. Chew, *Antennas and Propagation Society International Symposium, 1995. AP-S. Digest*. Vol. 3. IEEE, 1995.
19. Fast and Efficient Algorithms in Computational Electromagnetics, W. Chew, E. Michielssen, J.M. Song, J.M. Jin (Eds.), Artech House, Inc., Norwood, MA, USA, 2001.
20. A wideband fast multipole method for the Helmholtz equation in three dimensions, H. Cheng, W.Y. Crutchfield, Z. Gimbutas, L.F. Greengard, et al., J. Comput. Phys. 216 (1) (2006) 300-325.
21. RPYFMM: Parallel adaptive fast multipole method for Rotne–Prager–Yamakawa tensor in biomolecular hydrodynamics simulations, Guan, W., *et al. Computer physics communications* 227 (2018): 99-108.
22. An n log n parallel fast direct solver for kernel matrices, Chenhan, D. Yu, William B. March, and George Biros, *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017.

BACKUP SLIDES

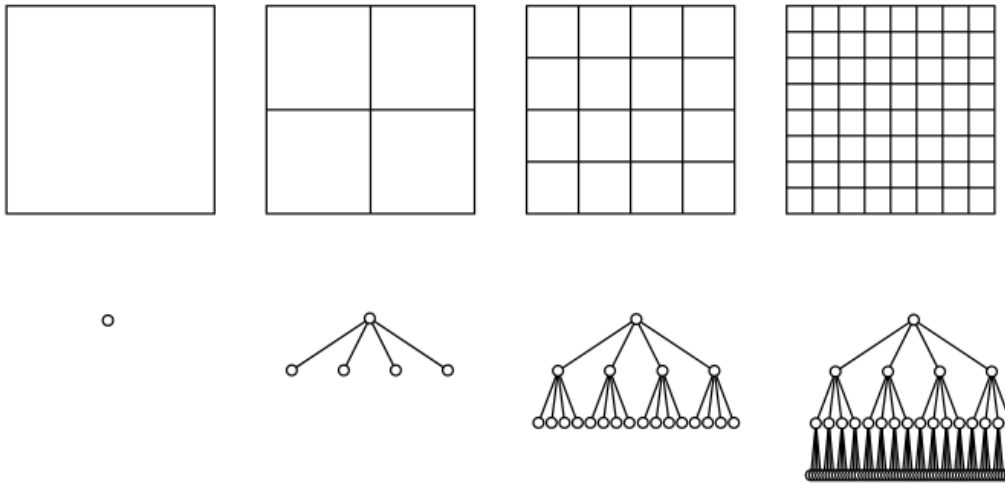
Single Level FMM

- Single level FMM works well for uniform charge distribution.
- It demonstrate the principle and the procedure of FMM.
- It is easier to understand.
- For more complicated charge distribution, multiple level fast multipole algorithm (FLFMA) is better, which will be explained later.

- Four steps procedure of FMM
  1. Decompose the space hierarchically into boxes of tree structure
  2. Going up the tree to calculate the multipole expansions in all the boxes
  3. Going down the tree to calculate the local expansions in all the boxes
  4. Calculate the potential/field inside the lowest level boxes

Single Level FMM: Domain Decomposition

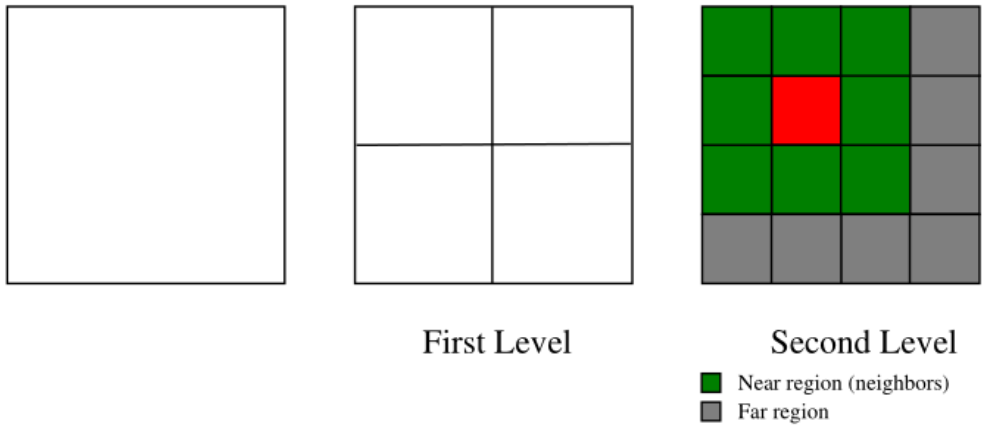
- Take a 2D problem as an example. 3D problems are treated in the same way
- Enclose the whole domain under study inside a large square box, the root box
- Cut the root box into four boxes with equal size. These are the boxes at level 1 and they are the child boxes of the root box.
- Generate child boxes for each level one boxes in the same way. These are the level 2 boxes.
- Keep cutting until the number of particles in each box at the finest level is smaller than a predetermined number S. Note: S only affects the efficiency, not the accuracy, of the algorithm.



Hierarchical tree structure of boxes

Single Level FMM: Near Region and Far Region

- For each box in the second or finer level, we can define the near region and the far region of the boxes.
- A box itself and all the boxes that touches it form the near region
- All the other boxes form the far region
- We will calculate the multipole expansion for each box, which is valid in the far region of the box

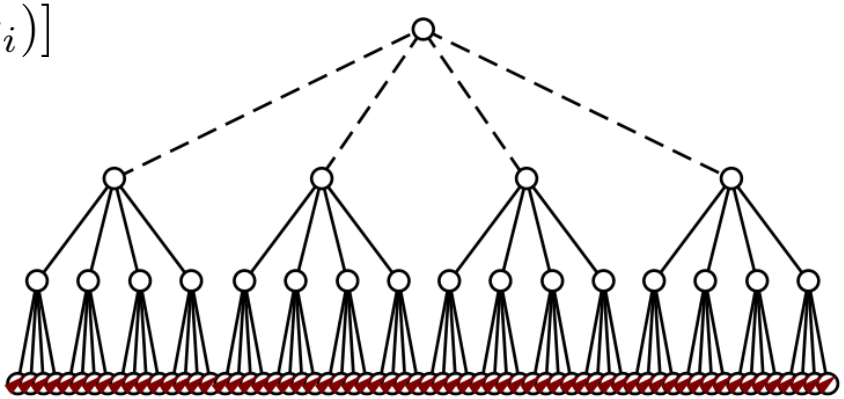


Single Level FMM: Multipole Expansion

- Multipole expansion are calculated from the finest level boxes to the coarsest level boxes (going up the tree).
- The multipole expansion for the finest level boxes are calculated from the particles inside the boxes.
- One needs to find an representation of the kernel function, which:
  1. converges fast and
  2. separate the sources and the objects $\phi(\mathrm{r}, \mathrm{r}') = M(\mathrm{r}) \circ f(\mathrm{r}')$
- Once we can separate the sources and the objects, the summation on the source particles can be calculated only on the source part:

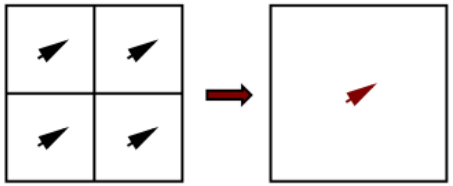
$$\phi = \sum \phi_i(\mathrm{r}_i, \mathrm{r}') = [\sum M(\mathrm{r}_i)] \circ f(\mathrm{r}')$$

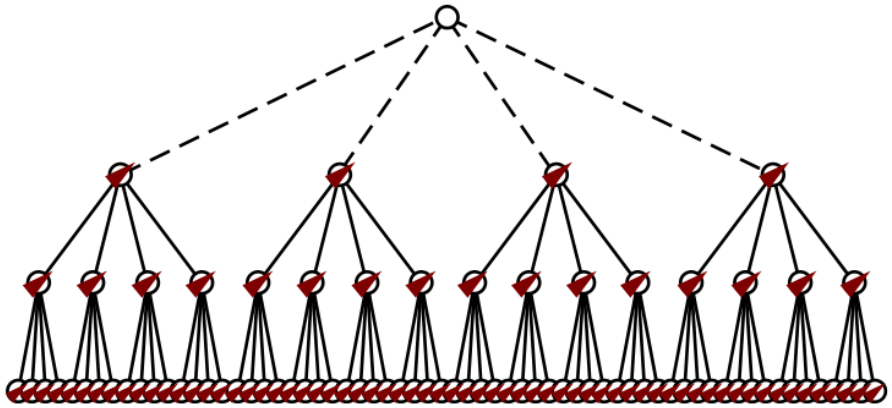
  ϕ is valid for arbitrary r' in the far region.
- The multipole expansion is $\mathrm{M} = [\sum M(\mathrm{r}_i)]$



Single Level FMM: Multipole Expansion

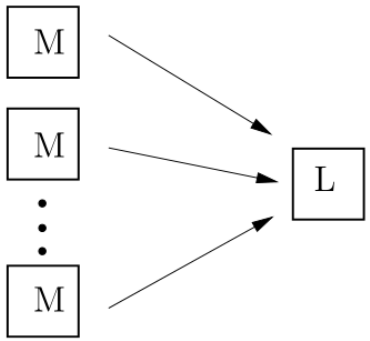
- For the boxes in the coarser level, the multipole expansion is constructed from the multipole expansions of their child boxes.
- We need to find an operator that moves the multipole expansion from one place to another place.
- Then we can move the multipole expansions at the center of the child boxes to the center of the parent box. Simply taking summations of these multipole expansions, we obtain the multipole expansion for the parent box.
- Going up the tree level by level, we can calculate the multipole expansions for all the boxes.





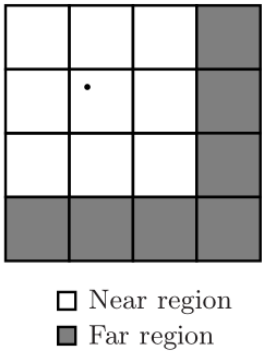
Single Level FMM: Local Expansion

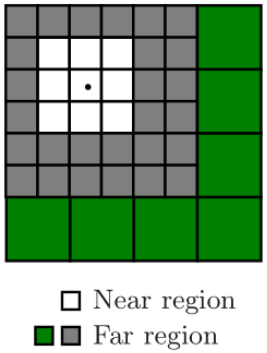
- The multipole expansion is valid for anywhere in the far region.
- The multipole expansion can be converted into a local expansion that is valid inside a box in the far region.
- For each box, we can convert the multipole expansions of the boxes in the far region of the box into local expansions inside the box and combine them into one local expansion.
- Here we need an operator that convert a multipole expansion at one place to a local expansion at the other place.
- This process goes from the top of the tree to the bottom of the tree.

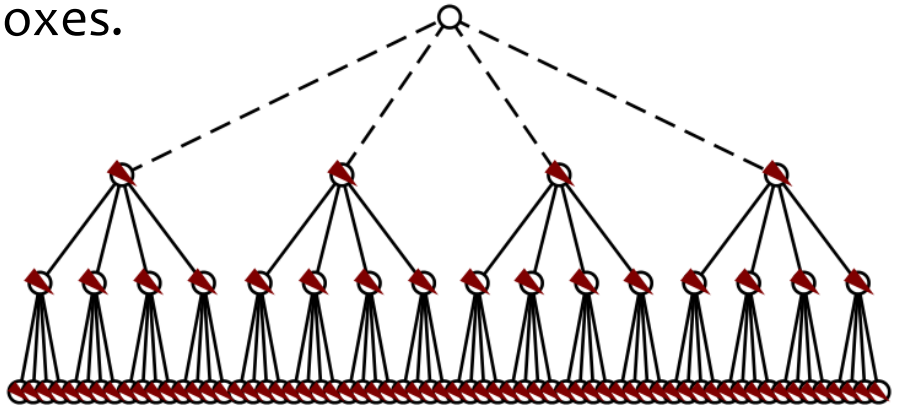


Single Level FMM: Local Expansion

- At the second level, we need to convert the multipole expansions of all the gray boxes into the box with a dot.
- At the third level, we only need to convert the multipole expansions of all the gray boxes into the boxes with a dot. Those of the green boxes has been converted into the parent box of the pointed box and can be transferred downwards to the pointed box.
- In this way, we limited the usage of the Multipole to Local expansion operator, which is the most time consuming part of FMM.
- Of course, we need an operator that moves the local expansion from one place to another place.
- Going downwards, we can calculate the local expansion of all boxes and translates them into the finest level boxes.

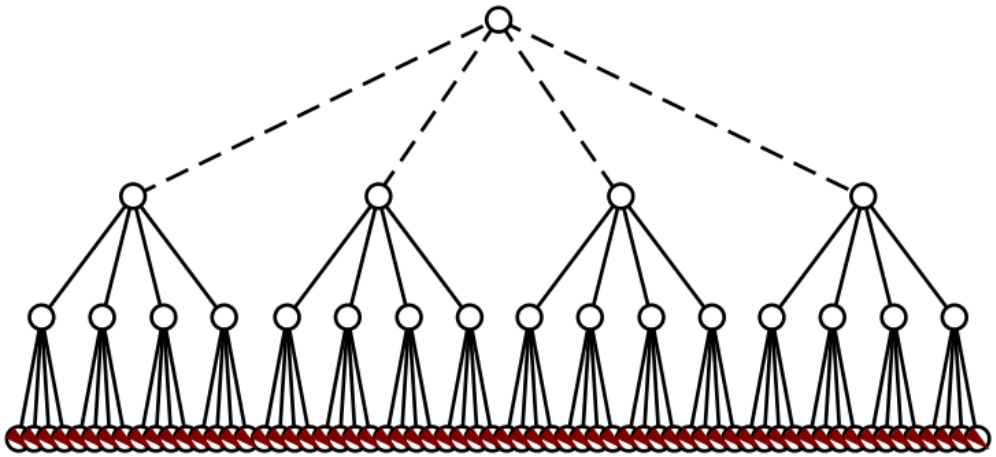






Single Level FMM: Evaluate the Kernel Function

- This step is done only in the finest level boxes (Childless boxes).
- The near region contribution is calculated using the kernel function directly.
- The far region contribution is calculated using the local expansion.
- Take summation of both contributions



Single Level FMM: An Implemetation

- There are various ways to construct your expansions
  - Using spherical harmonic functions [1,2]
    - The first FMM
    - Greengard and Roklin
    - https://pypi.python.org/pypi/pyfmmlib
  - Using Taylor expansion
    - Feng Zhao [3]
  - Using Cartesian tensor (Taylor expansion) for r^{-v} with any real v
    - H. Huang and B. Shanker [6]
  - Using differential algebra
    - H. Zhang and M. Berz [4,5]

- Let's take the Cartesian tensor based FMM as an example to see how to construct the multipole expansions and the local expansions.
- The Cartesian tensor based FMM is straightforward and the math may be less scary to beginners.