Fast Multipole Method for Multi-particle Simulations

He Zhang

ICAP'18

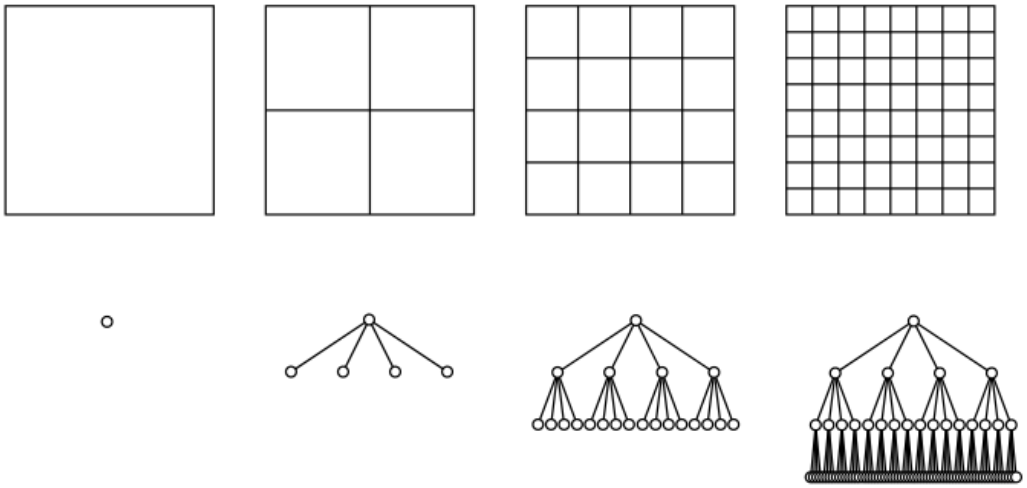
10/21/2018 Key West, FL, USA

Introduction of FMM

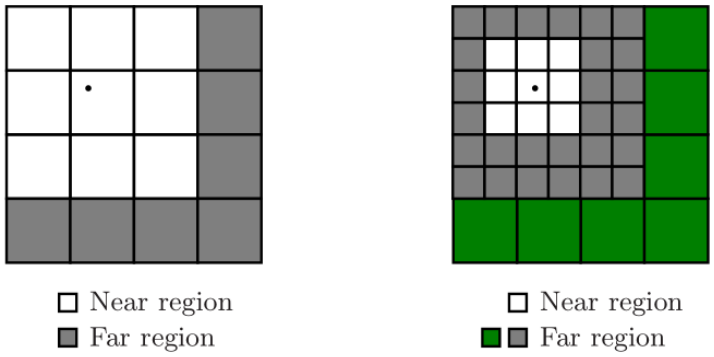
- A fast algorithm to calculate the interaction among *N* particles.
- Basic idea of FMM
  - the kernel can be represented by expansions that converges with distance.
  - Group the particles by their positions
  - Near region interaction is calculated directly by the pairwise formula
  - Far region interaction is calculated by the multipole/local expansion
- Property of FMM
  - Gridless
  - Scales $O(N)$ for non-oscillatory kernel with open boundary, and $O(N\log N)$ for oscillatory kernel with open boundary.
  - Accuracy is determined only by the order/rank of the expansions and the error can be strictly estimated and controlled.
  - Can be combined with other methods to solve boundary value problem.
- FMM is widely used in electric engineering, mechanic engineering, biophysics, *etc*. But not yet in accelerator/beam physics.

Single Level FMM

- Consider a simple case: Coulomb interaction between uniform distributed charges (2D for easier illustration)
- Group the particles: tree structure of boxes

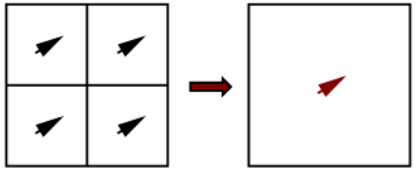


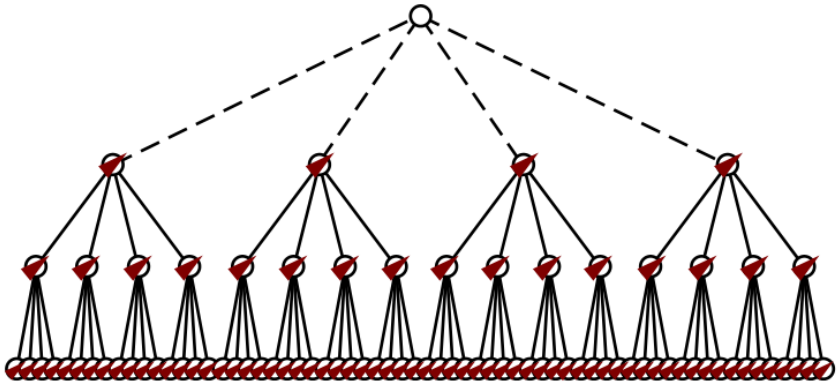
- Near region & far region



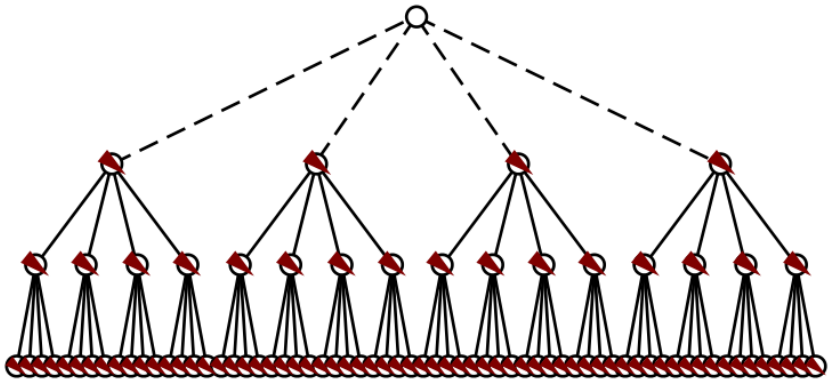
Single Level FMM

- Multipole expansion:
  - Converges in the far region
  - Bottom-up

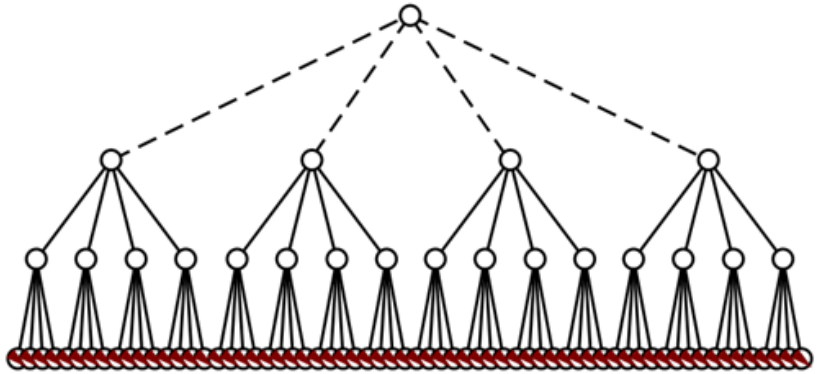




- Local expansion:
  - Convert all the multipole expansions in the far region into one local expansion



- Kernel evaluation:
  - Transfer all the local expansions to the bottom (Top-down)
  - Near region (formula) + far region (local expansion)



Single Level FMM

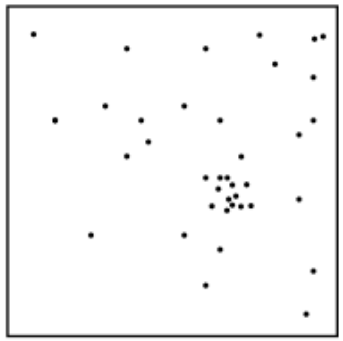
- Four steps procedure of FMM
  1. Decompose the space hierarchically into boxes of tree structure
  2. Going up the tree to calculate the multipole expansions in all the boxes
  3. Going down the tree to calculate the local expansions in all the boxes
  4. Calculate the potential/field inside the lowest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

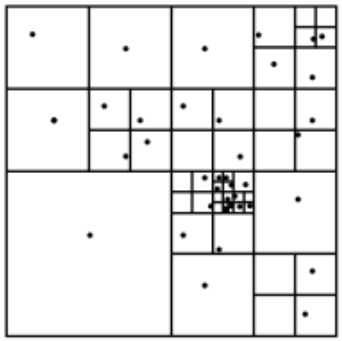
- How about non-uniform distributions?

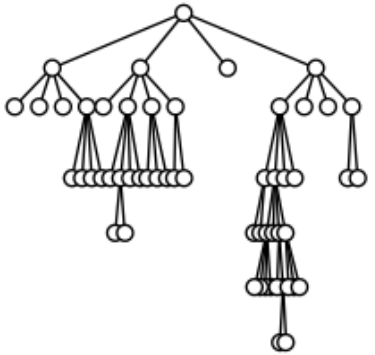
Multiple Level Fast Multipole Algorithm (MLFMA)

- For more complicated charge distribution, you need multiple level fast multipole algorithm (MLFMA). Still scales O(*N*).
- MLFMA follows the same procedure with the single level FMM.

- Decompose the domain until the number of particles inside each finest box is smaller than a predetermined number *S*. Again, *S* only affects the efficiency, not the accuracy.
- This time you get a partial tree. Smaller boxes are generated where the charge density is higher.



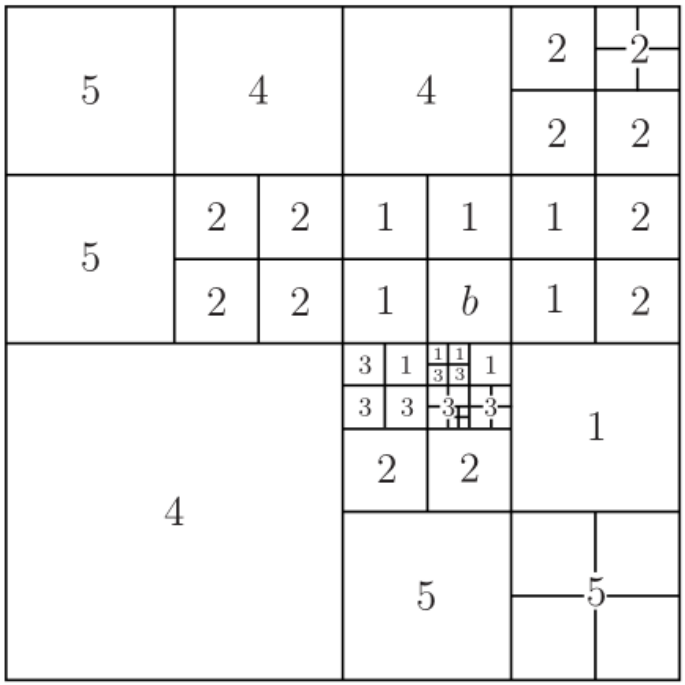




MLFMA: Relation of Boxes

- Boxes relations are more complicated.

- Consider a box *b*, there are five kinds of relations.
- 1: near region, as in single level FMM
- 2: far region, as in single level FMM
- 3: distance larger than the box size of 3, but smaller than the box size of *b*, new relation
- 4: distance larger than the box size of *b*, but smaller than the box size of 3, new relation
- 5: far region. In relation 2 with the ancestor boxes of *b*.



Relations		Operations
$c \in 1_b$	$b \in 1_c$	$C_c \to C_b$, $C_b \to C_c$
$c \in 2_b$	$b \in 2_c$	$M_c \to L_b$, $M_b \to L_c$
$c \in 3_b$	$b \in 4_c$	$M_c \to C_b$, $C_b \to L_c$
$c \in 4_b$	$b \in 3_c$	$C_c \to L_b$, $M_b \to C_c$
$c \in 5_b$	$b \in 5_c$	Do nothing

What do you need to construct FMM?

- Find the multipole expansion (P2M):
  - Separate the source and the observer: $\phi(\mathrm{r}, \mathrm{r}') = M(\mathrm{r}) \circ f(\mathrm{r}')$

    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 $\mathrm{M} = [\sum M(\mathrm{r}_i)]$
  - Converge fast.
- Transfer the multipole expansion to the parent box (M2M)
- Convert the multipole expansion into the local expansion (M2L)
  - Most time consuming operation!
- Transfer the local expansion to the child boxes (L2L)
- Calculate the kernel by the local expansion (L2P)
- Calculate the kernel by the formula (P2P)

Another two formulas for MLFMA

- Calculate the local expansion by source particles (P2L)
- Calculate the kernel by multipole expansion (M2P)

Construct Your 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 differential algebra
    - H. Zhang and M. Berz [4,5]

- Let's take the Cartesian tensor based FMM as an example .

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.

- Now we have the multipole expansion and the local expansion for a function *f*, as long as the high order gradients of $f(\mathbf{r})$ can be calculated.
- If $f(\mathbf{r}) = 1/r$, we have

$$\partial_i^{n_1} \partial_j^{n_2} \partial_k^{n_3} \left(\frac{1}{r}\right) = (-1)^n r^{-2n-1} \sum_{m_1=0}^{\lfloor \frac{n_1}{2} \rfloor} \sum_{m_2=0}^{\lfloor \frac{n_2}{2} \rfloor} \sum_{m_3=0}^{\lfloor \frac{n_3}{2} \rfloor} c(m,n) \begin{bmatrix} n_1 \\ m_1 \end{bmatrix} \begin{bmatrix} n_2 \\ m_2 \end{bmatrix} \begin{bmatrix} n_3 \\ m_3 \end{bmatrix} r^{2m} x^{n_1-2m_1} y^{n_2-2m_2} z^{n_3-2m_3}$$

$$\text{where } n = n_1 + n_2 + n_3, \text{ and } m = m_1 + m_2 + m_3$$

$$c(m,n) = (-1)^m (2n-2m-1)!! \text{ and } \begin{bmatrix} n \\ m \end{bmatrix} = \frac{n!}{2^m m! (n-2m)!}$$

Cartesian Tensor Based FMM: Formulas

- Particles to multipole expansion (P2M)

Having k particles inside a box, each having charge q_i and placing at r_i w.r.t. the center of the box, the multipole expansion up to n-th rank can be calculated as

$$\mathbf{M}^{(n)} = \sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \mathbf{r}_i^n .$$

- Multipole expansion to multipole expansion (M2M)

Given a multipole expansion of k sources around $\mathbf{r}_s$, the multipole expansion around the point $\mathbf{r}_{s'}$ can be expressed as

$$\mathbf{M}'^{(n)} = \sum_{m=0}^{n} \sum_{P(m,n)} \frac{m!}{n!} \mathbf{r}_{ss'}^{n-m} \mathbf{M}^{(m)}$$

where $\mathbf{r}_{ss'} = \mathbf{r}_{s'} - \mathbf{r}_s$ and $P(n,m)$ is the permutation of all partitions of n into sets $n-m$ and m.

Or it can be written component-wise as

$$M'^{(n)}(n_1,n_2,n_3) = \sum_{m_1=0}^{n_1} \sum_{m_2=0}^{n_2} \sum_{m_3=0}^{n_3} \frac{(n-m)!}{n!} \begin{bmatrix} n_1 \\ m_1 \end{bmatrix} \begin{bmatrix} n_2 \\ m_2 \end{bmatrix} \begin{bmatrix} n_3 \\ m_3 \end{bmatrix} \mathbf{r}_{ss'}^{m}(m_1,m_2,m_3) M^{(n-m)}(n_1-m_1, n_2-m_2, n_3-m_3)$$

Cartesian Tensor Based FMM: Formulas

- Multipole expansion to local expansion (M2L)

A multipole expansion at *s* can be converted into a local expansion at *o*, assuming they are far away enough:

$$\mathbf{L}^{(n)} = \sum^{\infty} \frac{1}{n!} \mathbf{M}^{\{m-n\}} : \widetilde{\nabla}^m \frac{1}{r_{so}},$$

where $\widetilde{\nabla}$ is the derivative w.r.t. $\mathbf{r}_s$ and $\widetilde{\nabla} = -\nabla$.

- Local expansion to Local expansion (L2L)

A local expansion $\mathbf{L}^{(n)}$ that exists in the domain Ω_o centered around $\mathbf{r}_o$ can be shifted to a subdomain centered at $\mathbf{r}_{o'}$ using

$$\mathbf{L}'^{(n)} = \sum_{m=n}^{\infty} \binom{m}{m-n} \mathbf{L}^{(m)} : \mathbf{r}_{oo'}^{m-n}$$

where

$$\mathbf{r}_{oo'} = \mathbf{r}_{o'} - \mathbf{r}_o \text{ and } \binom{m}{m-n} = \frac{m!}{(m-n)!n!}$$

Cartesian Tensor Based FMM: Formulas

- Potential calculation on particles using the local expansion (L2P)

the potential at any point $\mathbf{r}_i$ inside domain Ω_o centered around $\mathbf{r}_o$ can be obtained using

$$\phi(\mathbf{r}_i) = \sum_{n=0}^{\infty} \mathbf{L}^{(n)} : \boldsymbol{\rho}_i^n,$$

where $\boldsymbol{\rho}_i = \mathbf{r}_i - \mathbf{r}_o$.

Here are the two new formulas for MLFMA

- Calculate the local expansion from charges far away (P2L)

The potential inside a observation domain Ω_o, centered at $\mathbf{r}_o$, due to k sources that are far away enough can be expressed as

$$\mathbf{L}^{(n)} = \sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \nabla^n \frac{1}{r_i},$$

where $\mathbf{r}_i$ is the coordinate of the i^{th} source with charge q_i with respect to $\mathbf{r}_o$, and $\boldsymbol{\rho} = \mathbf{r} - \mathbf{r}_o$.

Cartesian Tensor Based FMM: Formulas

- Calculate the potential from a multipole expansion (M2P)

The potential at any point $\mathbf{r}_i$ far away from domain Ω_o centered around $\mathbf{r}_o$ can be obtained using

$$\phi(\mathbf{r}_i) = \sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \boldsymbol{\nabla}^n \frac{1}{\rho_i},$$

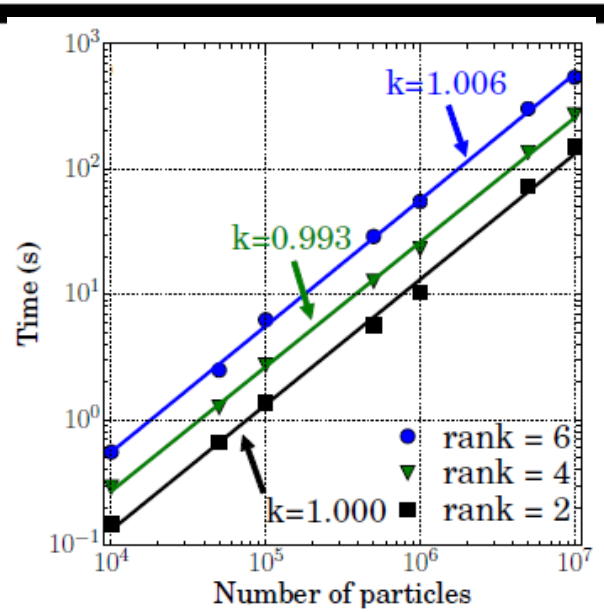
where $\boldsymbol{\rho}_i = \mathbf{r}_i - \mathbf{r}_o$.

- The field can be calculated by taking derivatives:

$$\mathbf{E}(\mathbf{r}) = -\nabla\phi(\mathbf{r}) = -\sum_{n=0}^{\infty} \nabla\mathbf{r}^n : \mathbf{L}^{(n)} \qquad \text{for } \mathbf{r}' > \mathbf{r}$$

$$\mathbf{E}(\mathbf{r}) = -\nabla\phi(\mathbf{r}) = -\sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \boldsymbol{\nabla}^{n+1} \frac{1}{r} \qquad \text{for } \mathbf{r} > \mathbf{r}'$$

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

FMM for Arbitrary Non-Oscillatory 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 any non-oscillatory kernel that you we 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]
  - The Cartesian tensor based FMM can also be extended for arbitrary non-oscillatory kernel, by H. Zhang & H. Huang, Let's see how to do it!

Cartesian Tensor based FMM for Arbitrary Non-Oscillatory Kernel

- Remember the Taylor expansion works for any non-oscillatory kernel function:

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

- If we can calculated high order gradients of $f(\mathbf{r})$, and replace all the gradients of 1/r in the formulas aforementioned by the gradients of $f(\mathbf{r})$, we are done!

- We do NOT know the kernel function before hand, so symbolic calculation of the gradients are very difficult.

- But, noting that in all the three cases we need to calculate the gradients, the value of **r** is known, the gradients actually can be calculated numerically/algorithmically by a tool that accelerator/beam physicists are very familiar with: Truncated Power Series Algebra (TPSA) or Differential Algebra (DA)

- Now we are really DONE!

Parallelization of FMM

- Parallelization of MLFMA is very challenging due to the unbalanced partial tree.
- Fortunately, great efforts have been put to solve this problem, with great achievements.
- Now MLFMA parallelizes well on thousands or even more of nodes
- Some libraries:
  - Pyfmmlib: Parallel via OpenMP (shared memory)
    - Python interface with Fortran code underneath.
    - https://pypi.python.org/pypi/pyfmmlib
  - 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
- 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]

Boundary Value Problem

- Boundary value problem (BVP) can be solved by boundary element method (BEM) [14].
- The problem can have Dirichlet boundary condition, Neumann boundary condition or a mixed boundary condition of them.
- We need to derive the Boundary Integral Equation (BIE):

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

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

- Discrete the BIE on the boundary, we get

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

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.

- FMM is used to accelerate the calculation of the left part of the equation.

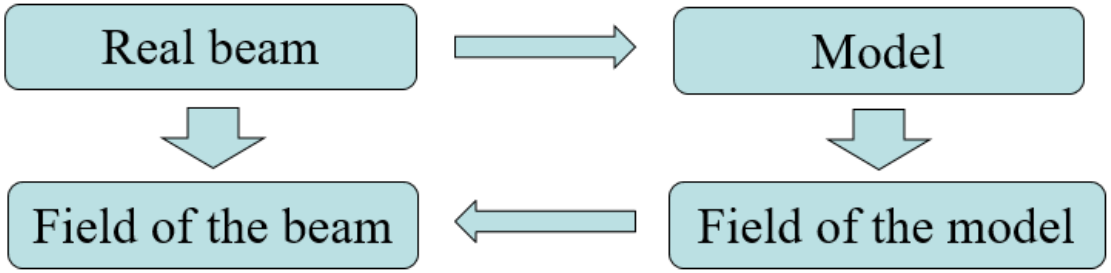
- Solving the linear system iteratively is very time consuming, so that BEM had been buried in dust for decades until MLFMA was implemented to accelerate the calculation.
- Note: Even for electrostatic BVP, Coulomb kernel is not enough for BEM. Kernel independent FMM needs to be used.
- FastBEM software: http://www.yijunliu.com/

Use FMM in Accelerator Study

- Particle-In-Cell (PIC) is the dominating algorithm of multi-particle simulation for accelerator studies.
- FMM could be a complementary algorithm
- FMM is gridless:
  - Works for any charge distribution and any geometry.
  - No redundant calculation on the grid points in free space.
  - Near region is calculated accurately by formula.
- Use a library is possible.
- But make sure you understand what it calculates and revise it to fit your problem if necessary!

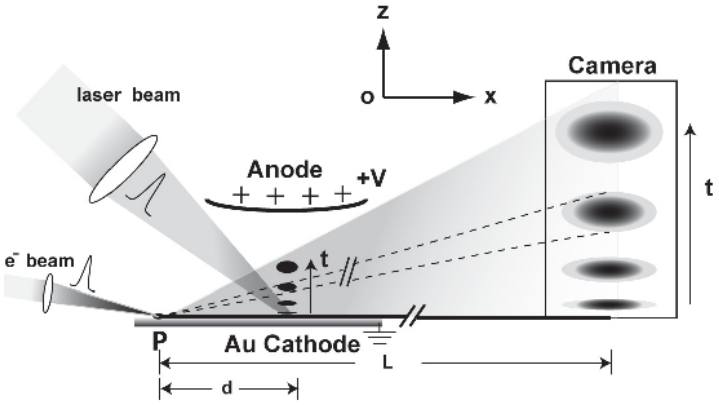
Use FMM in Accelerator Study

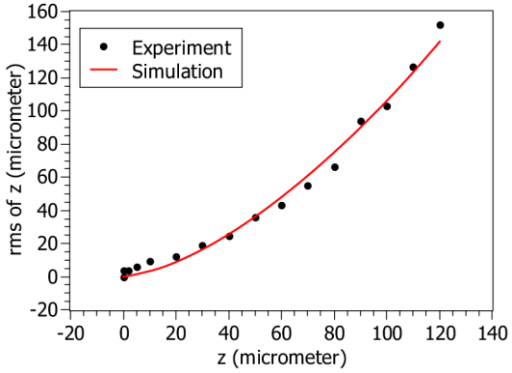
- Is FMM noisy?

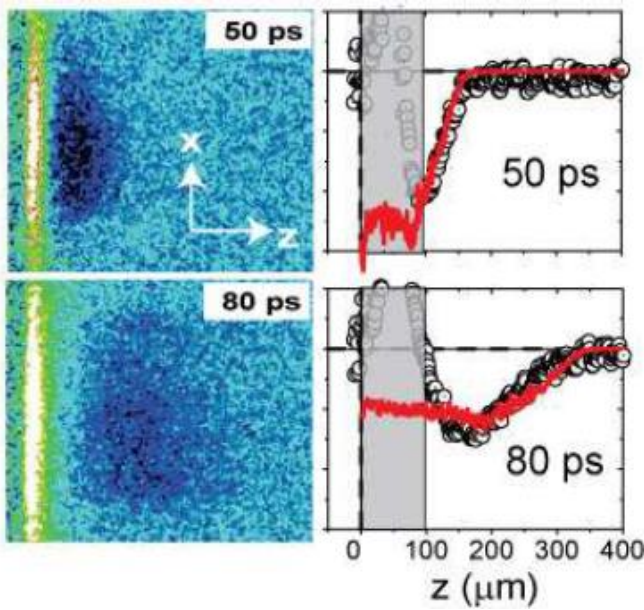


- There are two steps :1. Make the model of the beam; 2. Calculate the field of the model.
- Using PIC, the two steps are mixed. Even if you use point charges to model the beam, they are distributed to the grid when you calculate the field.
- FMM calculates the field of your model **honestly**.
- If you use point charges, noises are expected. If you believe the beam is smooth and the noises should be avoided, you need to use a different model (kernel function). For example, use small spherical Gaussian bunch instead of point charges or set a cut off distance for your point charge potential/field calculation.

Use FMM in Accelerator Study: An Example

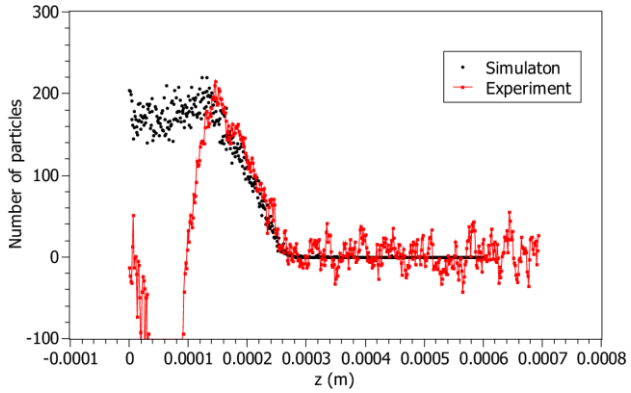


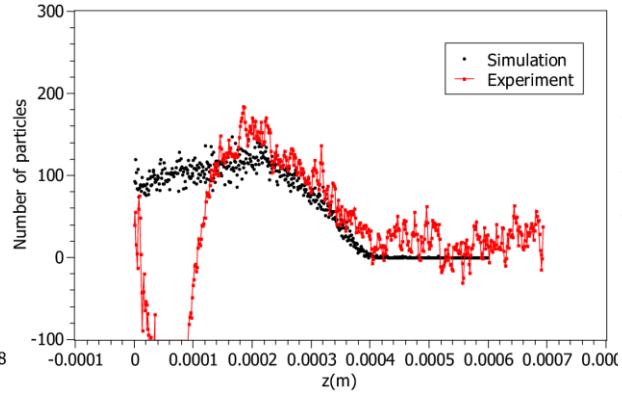


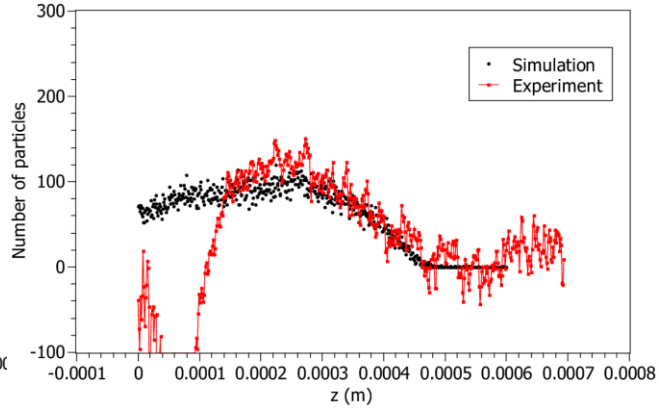


Simulation of the Photoemission process:

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







Use FMM in Accelerator Study: Other Attempts

- Prof. Martin Berz's group at Michigan State University:
  - space charge effect in photoemission process
- Prof. Bela Erldelyi's group at Northern Illinois University:
  - BVP
  - Electron cooling
  - http://niu.edu/beamphysicscode/
- Prof. Pavel Snopok at Illinois Institute of Technology:
  - Space charge effect in ionization cooling
- Prof. Yue Hao at Michigan State University:
  - Beam-beam effect
- Dr. He huang and Dr. He Zhang at Jefferson Lab:
  - FMM based algorithm development & implementation

- GSI - Space charge effect for FAIR (Talk yesterday afternoon.)

Oscillatory Kernel

- FMM can be used to solve electromagnetic problem in the frequency domain (Helmholtz equation). Computation scales O(*N*log*N*). [7,15, 20]
- Implementation is different with non-oscillatory kernel, but it shares the same idea: find the representation that converges fast and separate the sources and the objects.
- Problems with a general oscillatory kernel can be solved by
  - Kernel Independent FMM + Fast directional method, by L. Ying [16]
  - Black-box FMM + Fast directional method, by E. Darve [17]
- Boundary value problem can be solved by combining FMM with Combined Field Integral Equation (CFIE). [18]
  - https://sourceforge.net/projects/puma-em/?source=navbar
  - Parallel via MPI
- Electromagnetic problem in time domain remains an open question, although there are some works on this topic [19].

Summary

- FMM is a powerful tool to calculate the pairwise interaction between particles.
- Scales linearly with the number of particles.
- Grid-independent and suitable for complicated charge distribution and geometry.
- Can treat arbitrary non-oscillatory problem and oscillatory problem in frequency domain.
- Besides the open boundary problem, FMM can solve BVP if combined with BEM.
- High efficiency open source parallel FMM packages have been developed.
- Could be a useful tool for multiparticle simulations in accelerator study.


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

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

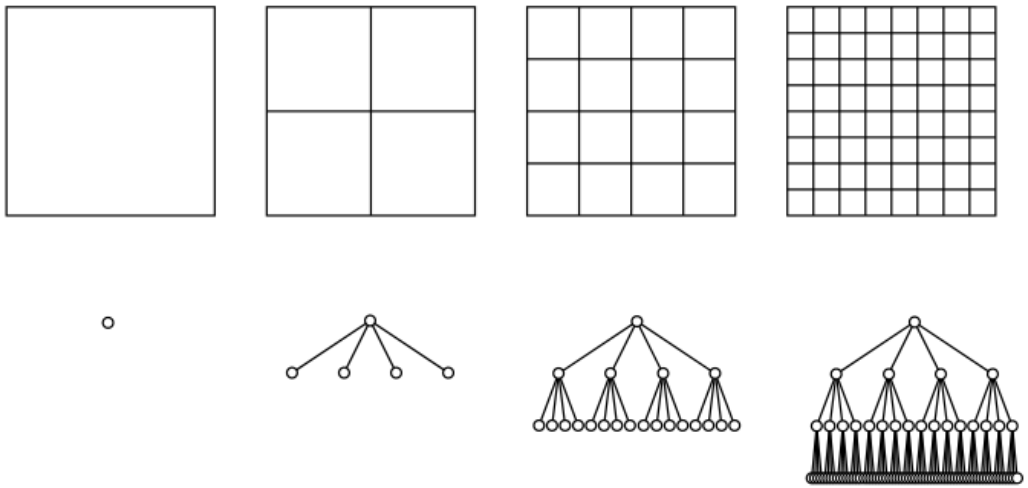
Single Level FMM

- 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

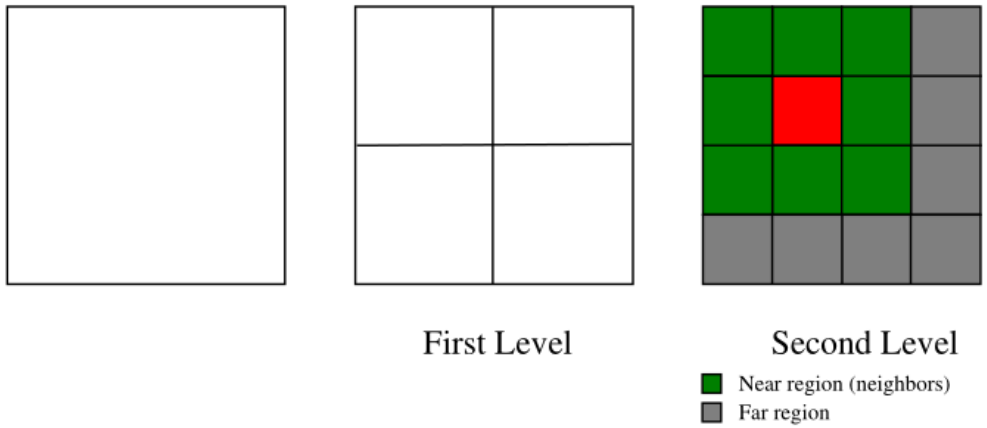
- Take a 2D problem as an example. 3D problems are treated in the same way
- Enclose the whole domain under study inside a large square box, the root box
- Cut the root box into four boxes with equal size. These are the boxes at level 1 and they are the child boxes of the root box.
- Generate child boxes 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

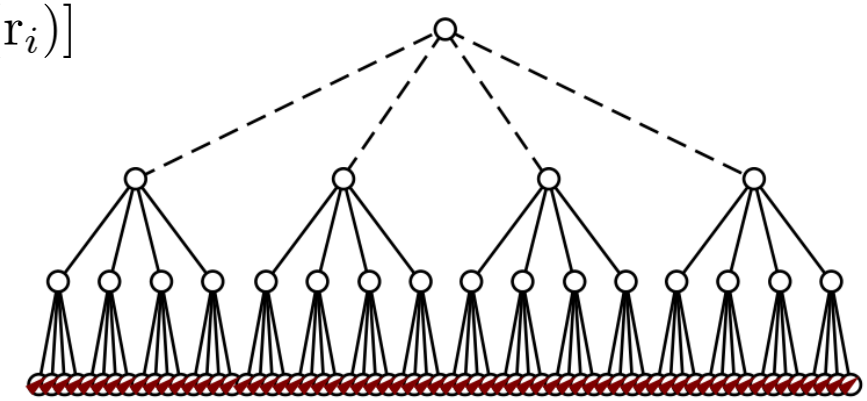


Single Level FMM: Multipole Expansion

- 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:

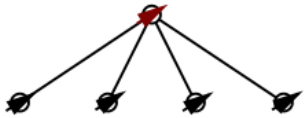
$$\phi = \sum \phi_i(\mathrm{r}_i, \mathrm{r}') = [\sum M(\mathrm{r}_i)] \circ f(\mathrm{r}')$$

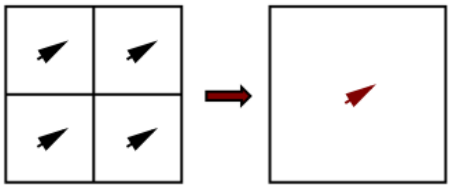
  ϕ is valid for arbitrary r' in the far region.
- The multipole expansion is $\mathrm{M} = [\sum M(\mathrm{r}_i)]$

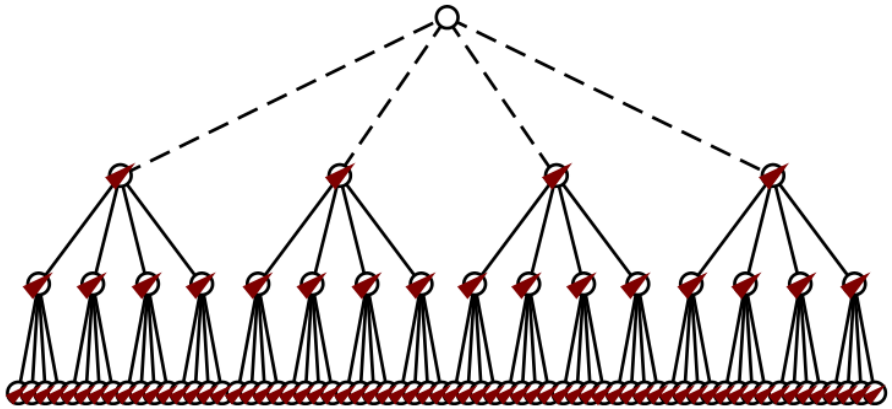


Single Level FMM: Multipole Expansion

- 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.

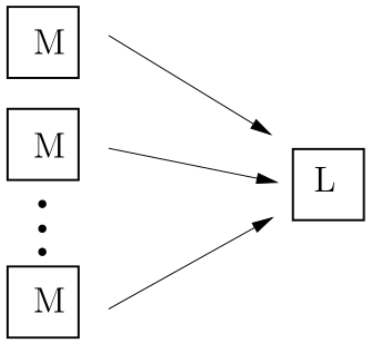






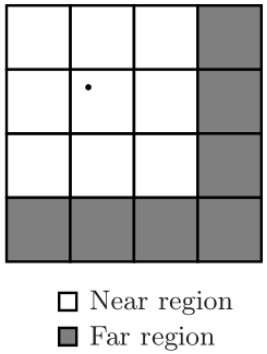
Single Level FMM: Local Expansion

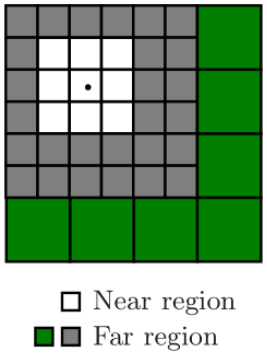
- The multipole expansion is valid for anywhere in the far region.
- The multipole expansion can be converted into a local expansion that is valid inside a box in the far region.
- For each box, we can convert the multipole expansions of the boxes in the far region of the box into local expansions inside 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.

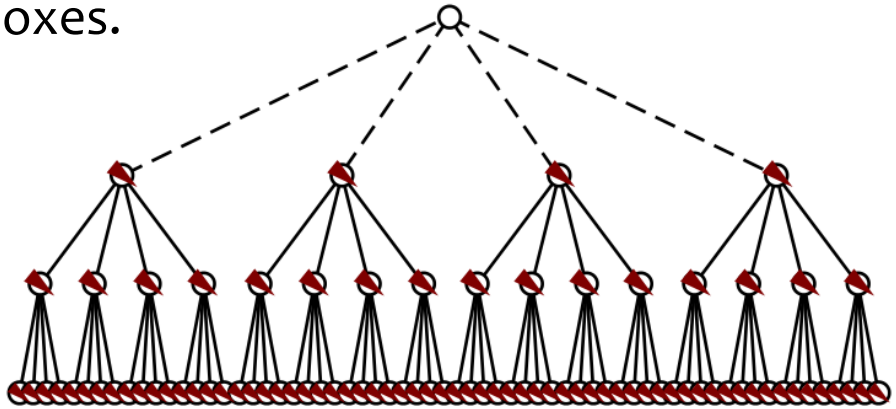


Single Level FMM: Local Expansion

- 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.

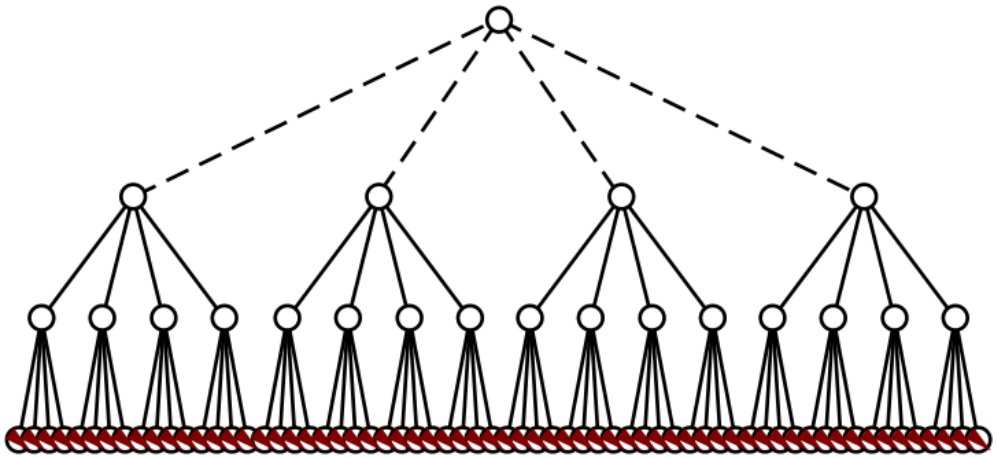






Single Level FMM: Evaluate the Kernel Function

- 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.