Fast Multipole Methods and their Implementation

Harald Hofstätter

Institute for Analysis and Scientific Computing
Vienna University of Technology

The Problem to be Solved

$$f(y) = \sum_{j=1}^{N} w_j k(y - x_j)$$

To be evaluated at $y \in \{y_1, \ldots, y_M\}$

- $k(\cdot)$ . . . kernel
- $x_j \in \mathbb{R}^d$ . . . sources, # = N
- $y_i \in \mathbb{R}^d$ . . . targets, # = M

Can be written as

$$\mathbf{f} = A \cdot \mathbf{w}, \qquad A_{i,j} = k(y_i - x_j)$$

Evaluation needs $O(NM)$ operations ($O(N^2)$ for $N \sim M$) in general

Structured Matrices

A dense, but depends only on $O(N + M)$ parameters

$\Rightarrow$ A has some "*structure*"

$\Rightarrow$ Faster evaluation of $\mathbf{f} = A \cdot \mathbf{w}$ likely to be possible

Most important example of a structured matrix:

$$F = \left[\omega_N^{ij}\right]_{i,j=0}^{N-1}, \qquad \omega_N = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N}}$$

$F \cdot \mathbf{x}$ by FFT ... $O(N \log N)$ operations

Structured Matrices

If x_i, y_i are *uniformly* spaced, e.g. $x_i = y_i = i \cdot h \in \mathbb{R}^1$:

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ a_{-1} & a_0 & \cdots & a_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{-N+1} & a_{-N+2} & \cdots & a_0 \end{bmatrix}, \quad a_i = k(i \cdot h)$$

A . . . *Toeplitz* matrix

$A \cdot \mathbf{w}$ by application of FFT . . . $O(N \log N)$ operations

Fast Multipole Methods

- Fast methods for *non*-uniformly spaced data (uniform distribution in some statistical sense necessary, not necessary for *adaptive* FMMs)
- Introduced by Rohklin & Greengard in 1987
- One of the *Top 10 Algorithms* (Computing in Science & Engineering 2000)
- Exploits *analytical* structure of kernel k, utilizes series expansions of k: Taylor, Multipole, Chebyshev, . . .)
- Expansions have to be truncated $\Rightarrow$ precision and complexity dependent on length p of expansions
- Error and computing costs can be estimated a-priori

Application: N-Body Problem

Simulation of stellar or molecular dynamics

$$f(y) = \sum_{j=1}^{N} w_j k(y - x_j)$$

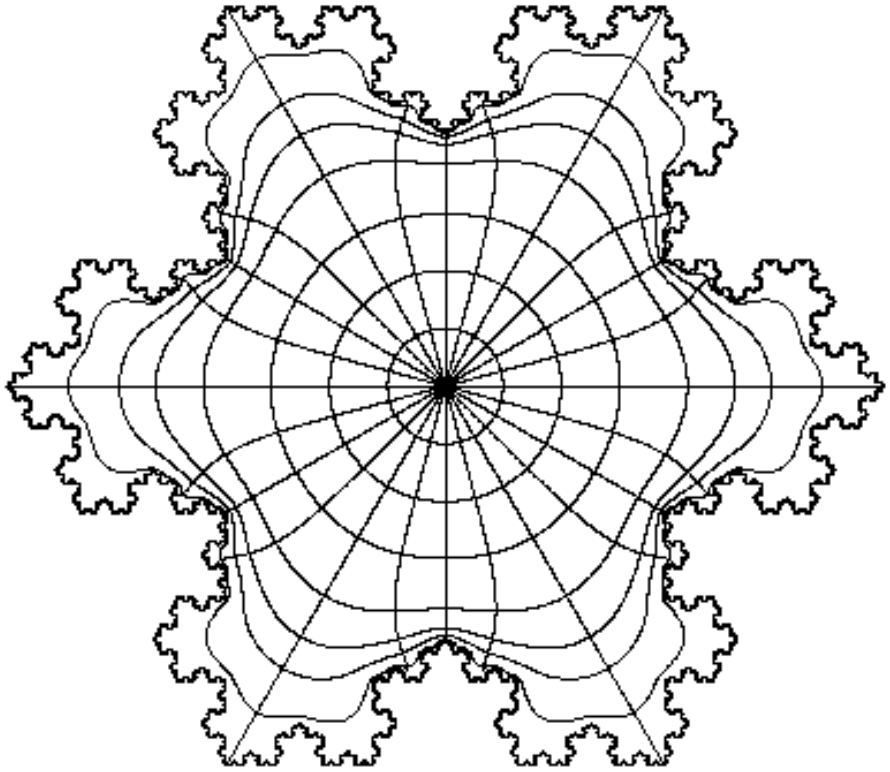
k . . . fundamental solution (Green's function) of Laplace's equation:

in $\mathbb{R}^2$: $k(y-x) = \log|y-x|$, in $\mathbb{R}^3$: $k(y-x) = \dfrac{1}{|y-x|}$

- Newtonian gravity potential at y due to masses w_j at x_j
- Coloumb's electrostatic potential at y due to charges w_j at x_j

Application: Conformal Mapping

L. Banjai and L.N. Trefethen: A Multipole Method for Schwarz-Christoffel Mapping of Polygons with Thousands of Sides



Koch snowflake polygon with 196608 vertices

Application: Conformal Mapping

Schwarz-Christoffel Formula:
f : unit disk $\rightarrow$ interior of polygon

$$f(z) = A + C \int^z \prod_{j=1}^{N} (\zeta - z_j)^{\alpha_j - 1} d\zeta$$

$\alpha_j \pi$ . . . interior angles of polygon
z_j . . . prevertices $\in$ unit circle,
have to be determined iteratively
Integral is evaluated numerically, $\log$ of integrand:

$$\sum_{j=1}^{N} (\alpha_j - 1) \log(\zeta - z_j)$$

$$k(y - x) = \log(y - x), \quad x, y \in \mathbb{C} \quad \text{(note: no } |\cdot|\text{)}$$

Application: Riemann Hypothesis

Numerical Verification of the Riemann Hypothesis:

"All non-trivial zeros of $\zeta(s)$ have real part $\frac{1}{2}$"

$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$ ... Riemann zeta function,
analytically continued to $\mathbb{C}\backslash\{0\}$

The *real* function

$$Z(t) = \exp(\mathrm{i}\theta(t))\zeta(\frac{1}{2} + \mathrm{i}t), \quad \theta(t) = \arg(\pi^{-\mathrm{i}t/2}\Gamma(\frac{1}{4} + \mathrm{i}\frac{t}{2}))$$

is efficiently computed by the Riemann-Siegel Formula

$$Z(t) = 2 \sum_{n=1}^{\lfloor\sqrt{t/(2\pi)}\rfloor} \frac{\cos(t \log n - \theta(t))}{\sqrt{n}} + \Phi(t) + R(t)$$

cos-sum ... most time consuming part

Application: Riemann Hypothesis

A.M. Odlyzko: Fast algorithms for multiple evaluations of the Riemann zeta function

$\cos$-sum $=$ real part of $\exp(-\mathrm{i}\theta(t))F(t)$, where

$$F(t) = \sum_{n=1}^{m} \frac{\exp(\mathrm{i}t \log n)}{\sqrt{n}}, \quad m = \lfloor \sqrt{t/(2\pi)} \rfloor$$

For numerical verification of RH:

$Z(t)$ resp. $F(t)$ needed for $t = T + j\delta,\ j = 0, \ldots, R-1$

$\rightarrow$ by FFT from

$$F(T + j\sigma) = \frac{1}{R} \sum_{k=0}^{R-1} u_k \omega^{jk}, \quad \omega = \mathrm{e}^{\frac{2\pi \mathrm{i}}{R}}, \quad u_k = \omega^k f(\omega_k),$$

$$f(z) = \sum_{k=1}^{m} \frac{a_k}{z - b_k}, \quad b_k = \mathrm{e}^{\mathrm{i}\delta \log k}, \quad a_k = \frac{\mathrm{e}^{\mathrm{i}T \log k}(1 - \mathrm{e}^{\mathrm{i}R\delta \log k})}{\sqrt{k}}$$

Divide-and-Conquer Algorithm:

$$T = \left[\begin{array}{cccc|cccc} a_1 & b_1 & & & & & & \\ b_1 & \ddots & \ddots & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & \\ & & b_{m-1} & a_m - b_m & & & & \\ \hline & & & & a_{m+1} - b_m & b_{m+1} & & \\ & & & & b_{m+1} & \ddots & \ddots & \\ & & & & & \ddots & \ddots & b_{n-1} \\ & & & & & & b_{n-1} & a_n \end{array}\right] + \left[\begin{array}{cc|cc} & & b_m & b_m \\ & & & \\ \hline & & b_m & b_m \\ & & & \end{array}\right]$$

$$= \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_m v v^T, \quad v = [0, \ldots, 0, 1, 1, 0, \ldots, 0]^T$$

Tridiagonal Eigenproblem

Eigendecompositions: $T_1 = Q_1 \Lambda_1 Q_1^T, \quad T_2 = Q_2 \Lambda_2 Q_2^T$

$$T = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left(\begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix}$$

i.e., T similar to

$$D + \rho u u^T$$

Characteristic polynomial of $D + \rho u u^T$:

$$\chi(\lambda) = 1 + \rho \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda}$$

Tridiagonal Eigenproblem

Eigenvector q_j for eigenvalue λ_j:

$$q_j = s_j(D - \lambda_j I)^{-1}u, \quad s_j = 1/\|(D - \lambda_j I)^{-1}u\|_2$$

Q often not needed explicitely, only $y^T = x^T Q$:

$$y_j = s_j \sum_{i=1}^{n} \frac{x_i u_i}{d_i - \lambda_j}$$

Combination of rapid solutions of eigen-sub-problems for $D + \rho u u^T$
$\Rightarrow$ Rapid solution of original eigenproblem for T

FFT for Non-equispaced Data

$x_k = 2\pi k/N$ . . . uniform, $y_j \in [0, 2\pi)$ . . . non-uniform

- $f(x)$ sampled at x_k, $f(x_k) = f_k$, find $g_j = f(y_j)$
- $f(x)$ sampled at y_j, $f(y_j) = g_j$, find $f_k = f(y_k)$

$$f(x) = \sum_{n=0}^{N-1} c_n e^{inx}, \qquad c_n = \frac{1}{N} f_k e^{-inx_k}$$

forward DFT: $(f_k) \to (c_n)$
inverse DFT: $(c_k) \to (f_k)$
} $O(N \log N)$ operations using FFT

Goal: Solve interpolation problem

$$(c_n) \rightleftarrows (g_j)$$

with the same efficiency

FFT for Non-equispaced Data

$$(g_j) = K \cdot (f_k)$$

$$K_{j,k} = \frac{1}{N} \sum_{n=0}^{N-1} e^{-nx_k} e^{iny_j} = \frac{1}{N} \frac{e^{iNy_j} - 1}{e^{i(y_j - x_k)} - 1} = F_j G(y_j - x_k)$$

$$F_j = \frac{e^{iNy_j}}{N}, \qquad G(t) = \frac{1}{e^{it} - 1} = -\frac{1 + i\cot(t/2)}{2}$$

$\Rightarrow$ $(g_j) = K \cdot (f_k)$ using FMM with kernel $\cot(\frac{y_j - x_k}{2})$

$$\cot\frac{t}{2} = \frac{2}{t} - 2\sum_{m=1}^{\infty} \frac{|B_{2m}|}{(2m)!} t^{2m-1}, \quad |t| < 2\pi$$

B_m . . . Bernoulli numbers

$$\cot(\frac{y_j - x_k}{2}) = \frac{2}{y_j - x_k} + f(y_j - x_k)$$

Kernel $f(t)$ *analytical* at 0

Kernel analytical at 0

$$f(y) = \sum_{j=1}^{N} w_j k(y - x_j)$$

Special case:

$$k(y - x_j) = \sum_{\ell=0}^{\infty} a_\ell (y - x_j)^\ell, \quad |y - x_j| \leq 2,$$

e.g. for $y, x_j \in$ unit disk

Idea: Truncate series and translate:

$$\sum_{\ell=0}^{p} a_\ell (y - x_j)^\ell \quad \mapsto \quad \sum_{\ell=0}^{p} b_{j,\ell} y^\ell$$

Then

$$f(y) \approx \sum_{\ell=0}^{p} B_\ell y^\ell, \quad B_\ell = \sum_{j=1}^{N} w_j b_{j,\ell}$$

B_ℓ available $\Rightarrow$ approximation for $f(y)$ . . . $O(p)$ operations

Multipole Expansion

$k(z)$ *singular* at $z = 0$,

but $k(y - x_j)$ can be expanded into a *power series* in $1/y$

Example:

$$\log(y - x) = \log(y) - \sum_{\ell=1}^{\infty} \frac{1}{\ell} \left(\frac{x}{y}\right)^{\ell}, \qquad |y| > |x|$$

Then, if $|x_j| < r$, and $|y| > r$,

$$\sum_{j=1}^{N} w_j k(y - x_j) \approx a_0 \log y + \sum_{\ell=1}^{p} \frac{a_\ell}{y}, \quad a_\ell = \sum_{j=1}^{N} \frac{-w_j {x_j}^{\ell}}{\ell}$$

(*Multipole expansion* of $f(y)$)

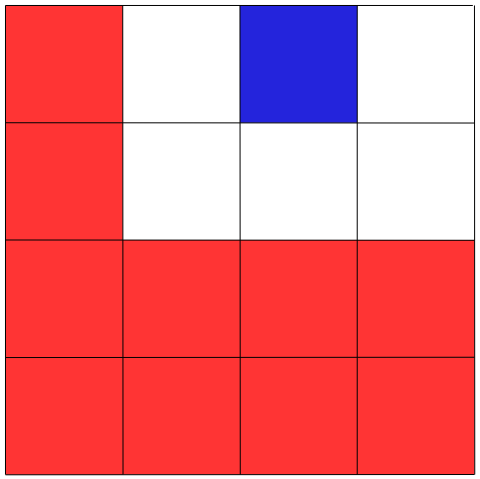
Error estimate:

$$err \leq \|\mathbf{w}\|_1 \left(\frac{1}{2}\right)^{p} \quad \text{for } |y| \geq 2r$$

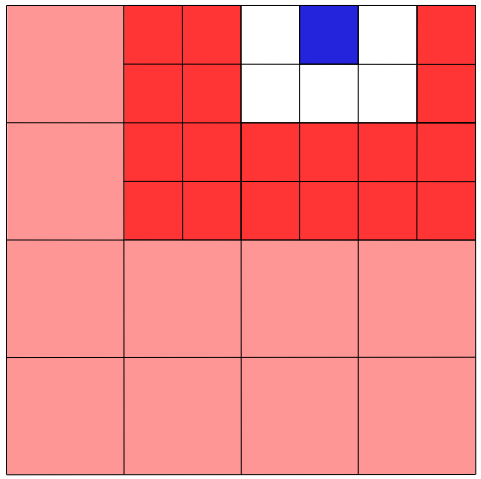
The $O(N \log N)$ Algorithm

Top-down traversal of space hierarchy:

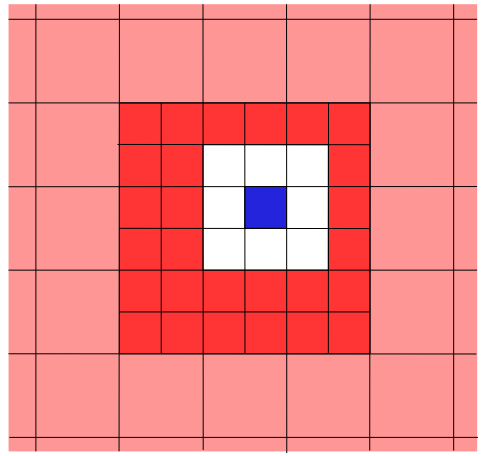
Creation of expansions around centers of each box
Work per level: $\sim Np$ (each particle contributes to p coefficients)



1st step



2nd step



general step

Work per level: $\sim 27Np$ (at most 27 boxes in each particle's interaction list)

$\sim \log N$ levels, in finest level: interactions of nearest neighbours are computed directly, Work $\sim 8N$ (in each of the $\sim N$ boxes ~ 1 particle witch ~ 8 nearest neighbours)

Fast Multipole Method

Two main phases of algorithm:

1. Setting up Hierarchical Data Structure

   $\Rightarrow$ For each box B in finest level: local expansion for field due to all particles outside B's near neighbours

   Local expansions obtained after upward and downward traversing of hierarchy (see below)

   Total work $O(N)$ (dependent on p)

2. Approximation of $f(y)$:

   Find Box B in finest level containing target y

   Evaluate local expansion of B

   Compute near neighbour interactions directly

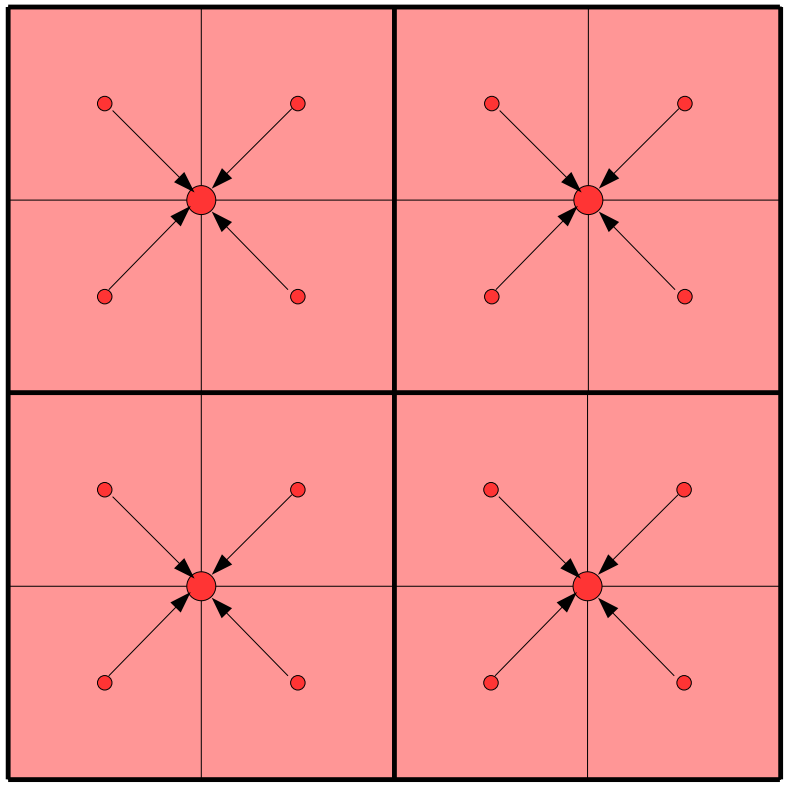
   Work $O(1)$ per target y (dependent on p)

FMM: Upward Pass

Generate multipole expansions for each box in every level

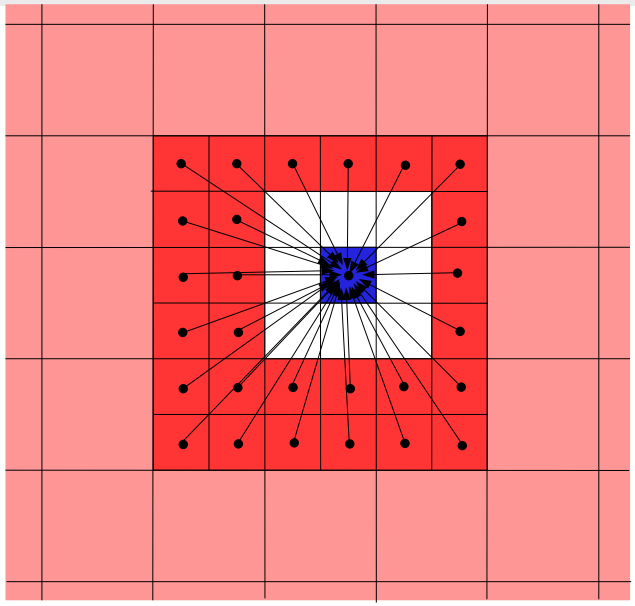
Only expansions of finest level are computed directly

Expansions of other levels by *merging* expansions from children boxes:

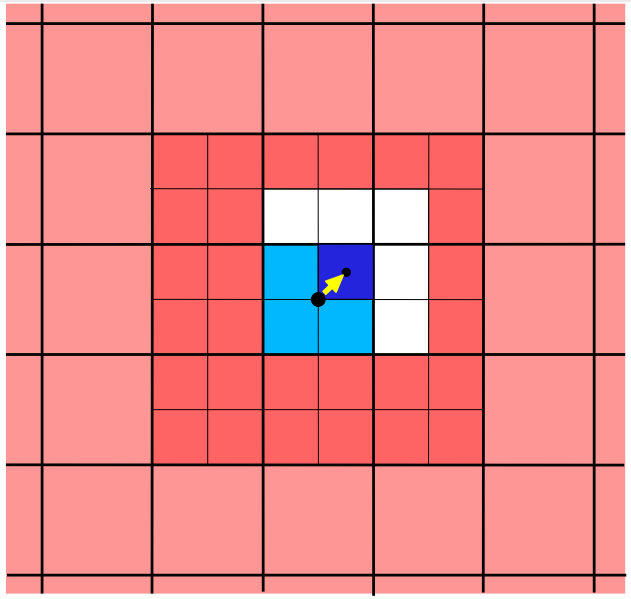


M2M

FMM: Downward Pass



M2L



L2L

For each box B in current level:
Convert multipole expansion of boxes of B's interaction list

$\Rightarrow$Local expansion in each box of each level

Expansions *shifted* to children's level and added to the children's expansions

$\Rightarrow$ For each box B in finest level: Local expansion for field due to all particles outside B's near neighbours
Near neighbour interactions computed directly

Software Development

Cooperation with H. Dachsel, J. Grotendorst
John v. Neumann Institute for Computing
Research Centre Jülich (Germany)

Molecular electronic structure calculations based on Continuous Fast Multipole Method (CFMM)

FMM: point charges
CFMM: charge distributions arising in Density Functional and Hartree Fock calculations

Speed up the existing code:

- Utilize state-of-the-art mathematical FMM-theory to minimize operation count
- Processor specific optimizations

Software Development

2D-FMM, 3D-FMM: have direct uses
(N-body problem, Boundary Integral Method, etc.)
Highly optimized, highly parallelized software available

1D-FMM (especially with kernel $1/(x-y)$*):* Like FFT,
only step in the solution of other numerical problems
(tridiagonal eigenproblem, non-equispaced FFT, etc.)
Highly optimized software currently not available
(as it is for FFT $\rightarrow$ FFTW)
On this we will focus our efforts

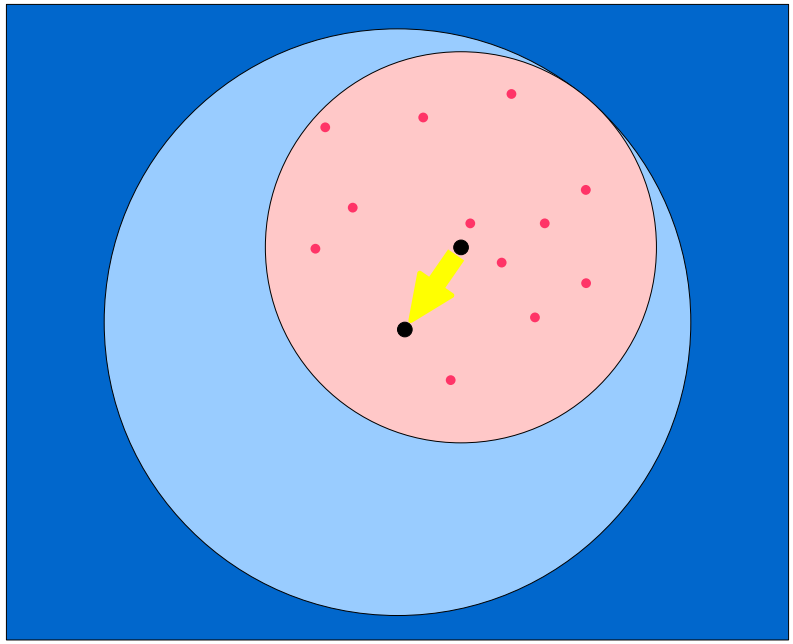
Goal:

FMMW, the Fastest Multipole Method in the West

Thank you for your attention!

Translation of Expansions

Multipole $\rightarrow$ multipole (M2M):

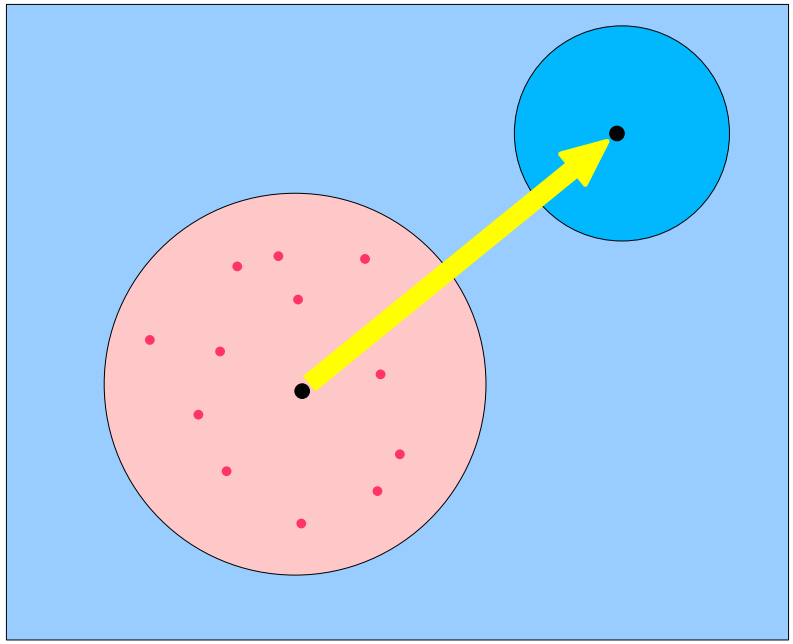


$$a_o \log(z - z_0) + \sum_{\ell=1}^{p} \frac{a_\ell}{(z - z_0)^\ell} \quad \mapsto \quad a_o \log(z) + \sum_{\ell=1}^{p} \frac{\tilde{a}_\ell}{z}$$

$(a_\ell) \mapsto (\tilde{a}_\ell)$ takes $\sim p^2$ operations (can be improved)

Translation of Expansions

Multipole $\rightarrow$ local (M2L):

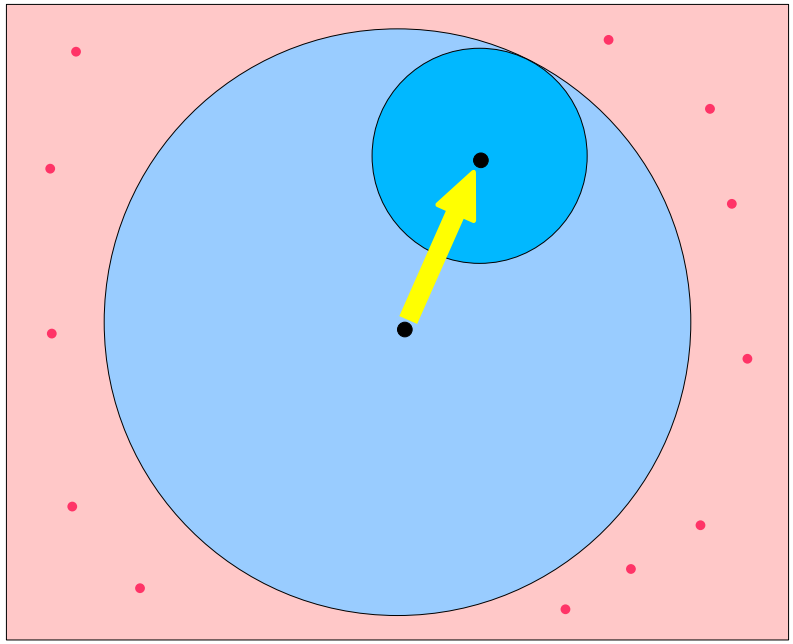


$$a_o \log(z) + \sum_{\ell=1}^{p} \frac{a_\ell}{z} \quad \mapsto \quad \sum_{\ell=0}^{p} b_\ell z^\ell$$

$(a_\ell) \mapsto (b_\ell)$ takes $\sim p^2$ operations (can be improved)

Translation of Expansions

Local $\rightarrow$ local (L2L):



$$\sum_{\ell=0}^{p} b_\ell (z - z_0)^\ell \quad \mapsto \quad \sum_{\ell=0}^{p} \tilde{b}_\ell z^\ell$$

$(b_\ell) \mapsto (\tilde{b}_\ell)$ takes $\sim p^2$ operations (can be improved)