Implementing Fast Multipole Methods with High Level Interpreted Languages

Srinath Kailasa

Department of Physics & Astronomy
University College London

September 9, 2020

Table of Contents

Table of Contents

Motivation - the N Body Problem

- e.g. Electrostatics, Gravitation
- $\{x_i\}_{i=1,..,N}$ Source Particles
- $\{y_j\}_{j=1,..,M}$ Target Particles

$$\Phi(y_j) = \sum_{i=1}^{N} K(x_i, y_j) q_i, \tag{1}$$

$$\text{where, } K(x, y) = \frac{1}{\|x - y\|} \tag{2}$$

- FMM reduces complexity from $O(N^2)$ to $O(N)$

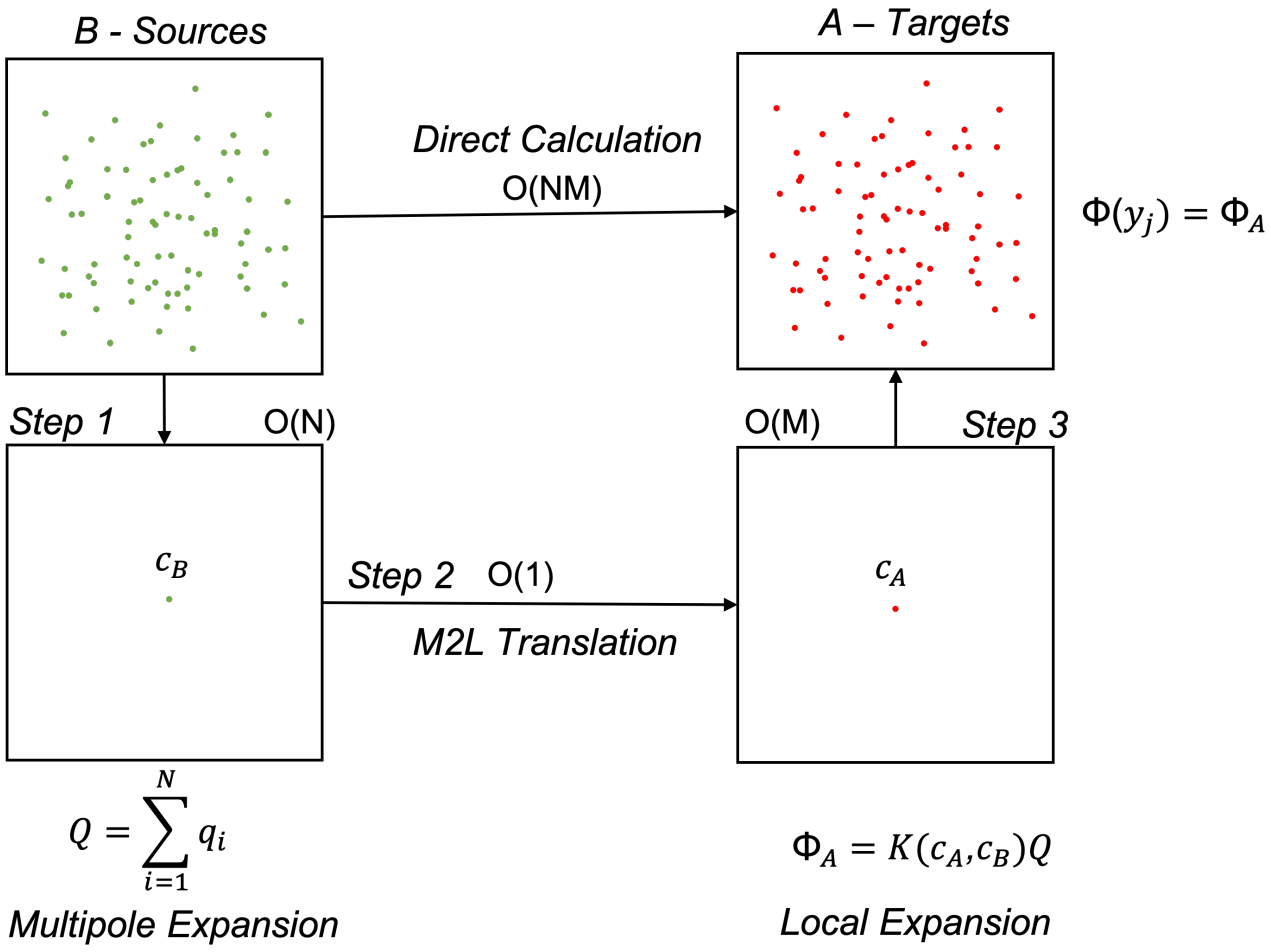
Motivation - Intuition

Galaxy A

Galaxy B

Motivation - Intuition

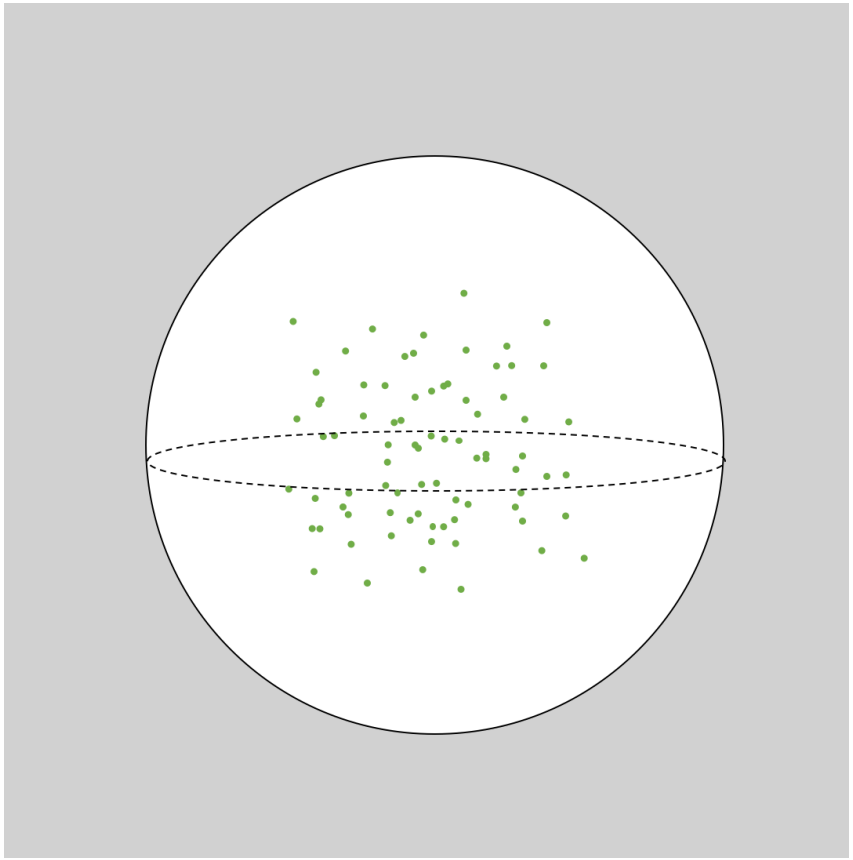
Motivation - Three Step Procedure



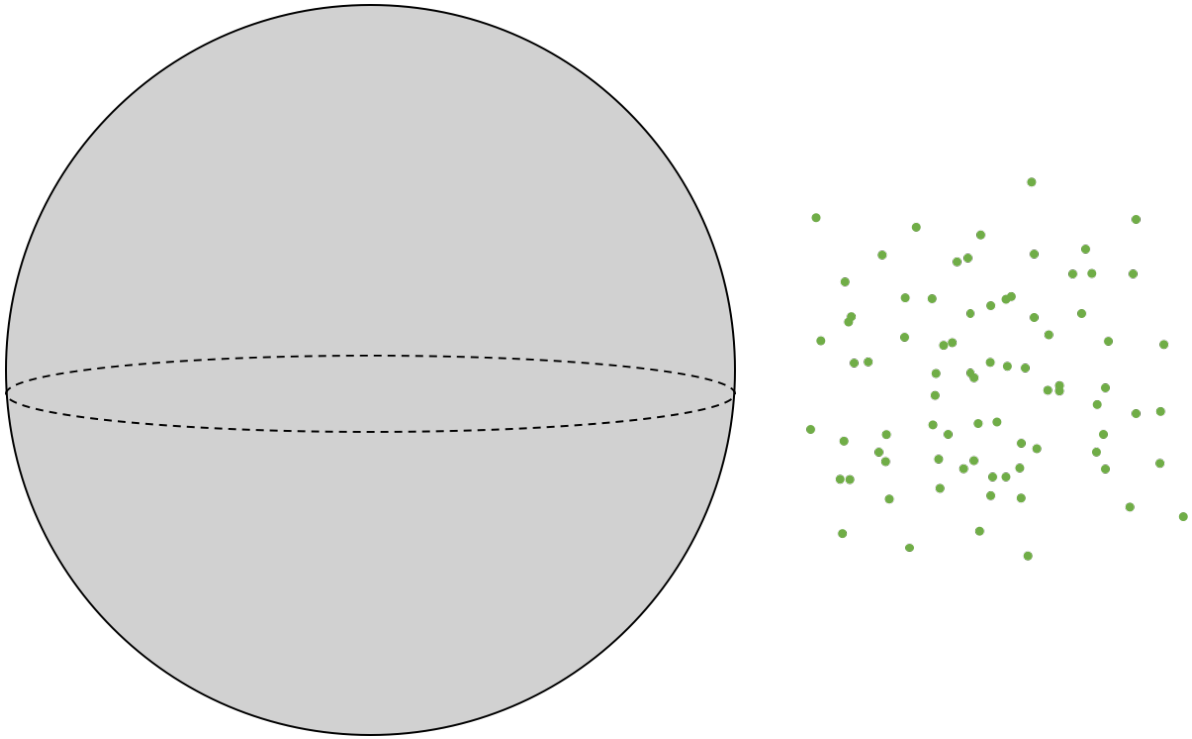
Analytic FMM - Concept

Idea: Use compressed representations of far field potentials to reduce complexity, in a recursive fashion

Analytic FMM - Multipole Expansion



Analytic FMM - Local Expansion

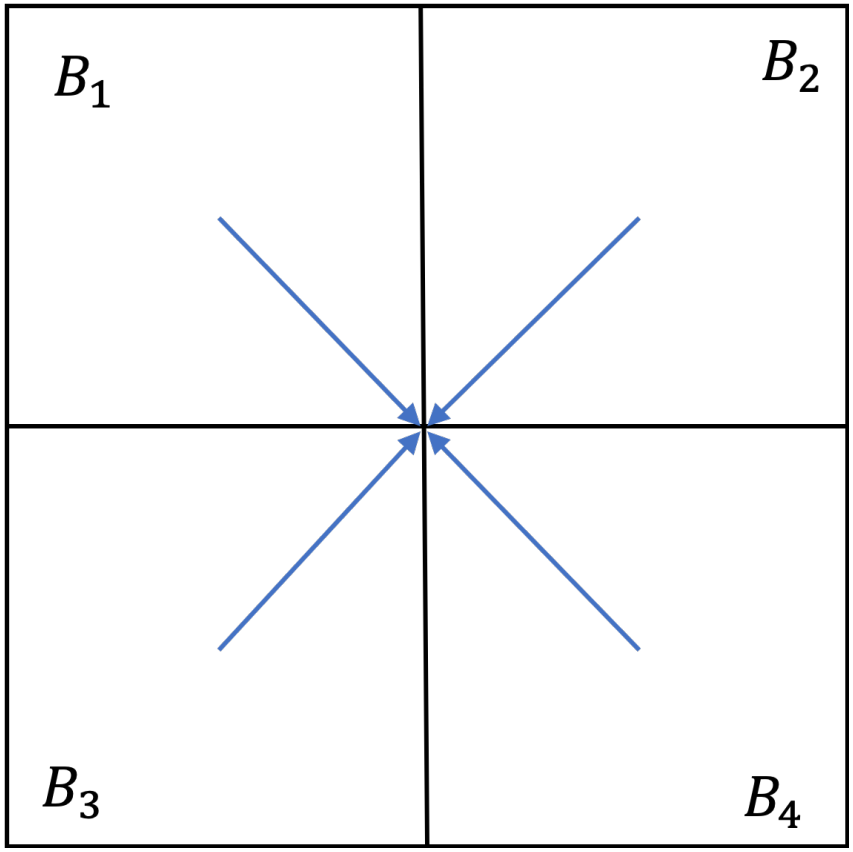


- For 3D Laplace kernel, can write multipole and local expansions using sph. harmonics, these can be truncated to required accuracy
- Exact operators exist for this kernel to shift between multipole and local expansion coefficients
- Exact bounds on error also exist, with respect to direct computation [1]

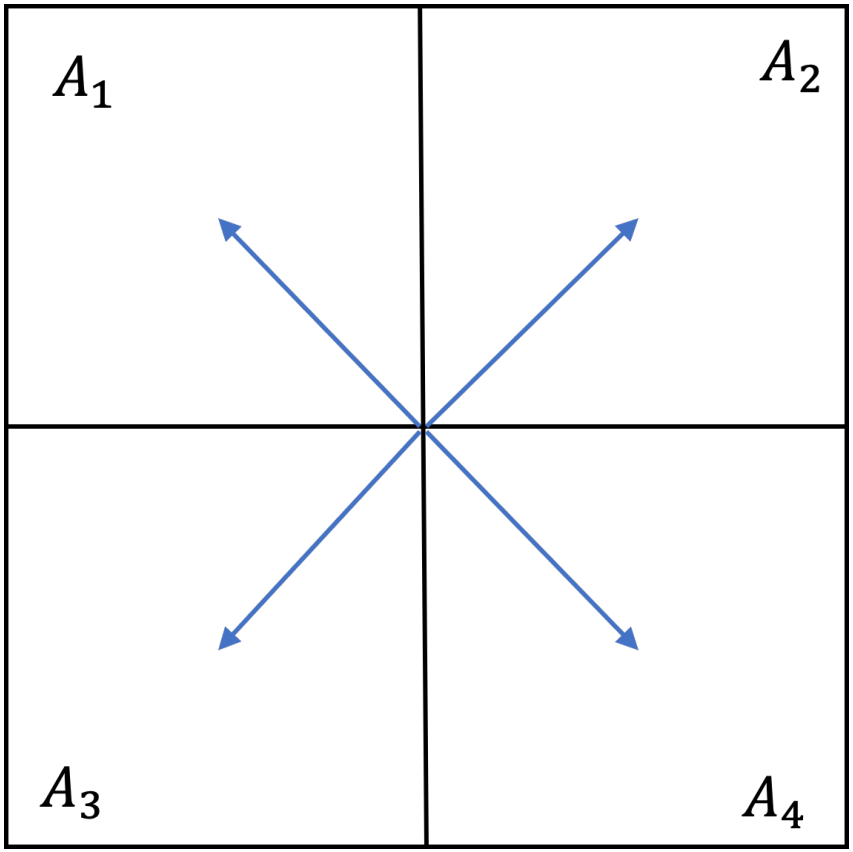
Analytic FMM - Motivating Problem

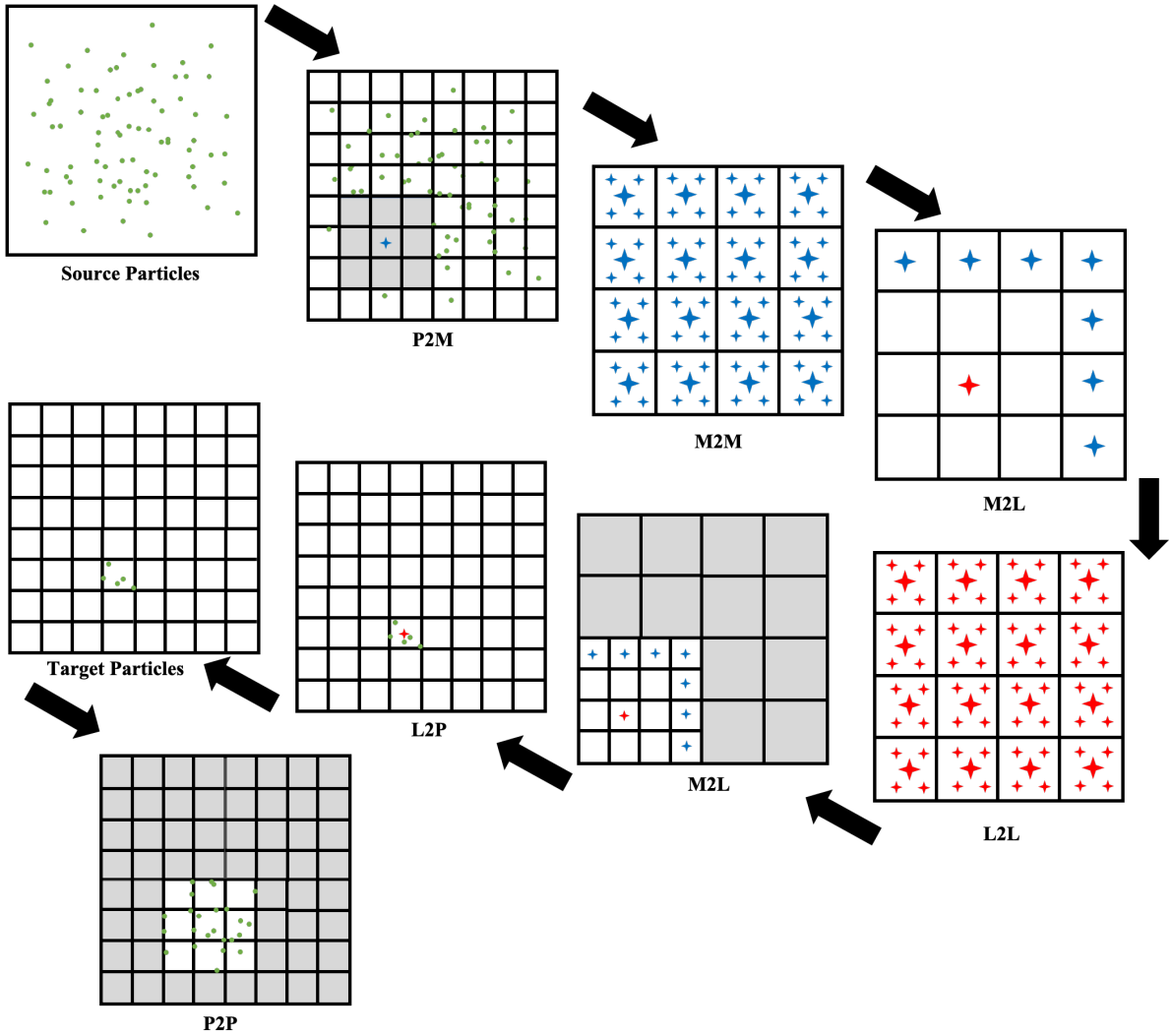
- Consider 2D Problem
- Domain, $\Omega = [0, 1] \times [0, 1]$
- Partition into recursively defined Quadtree
- Each level, l, partitioned into 4^l boxes
- Source and Target particles taken to be the same

Analytic FMM - Shifting Multipole Expansion



Analytic FMM - Shifting Local Expansion




Source Particles
P2M
M2M
M2L
L2L
M2L
L2P
Target Particles
P2P

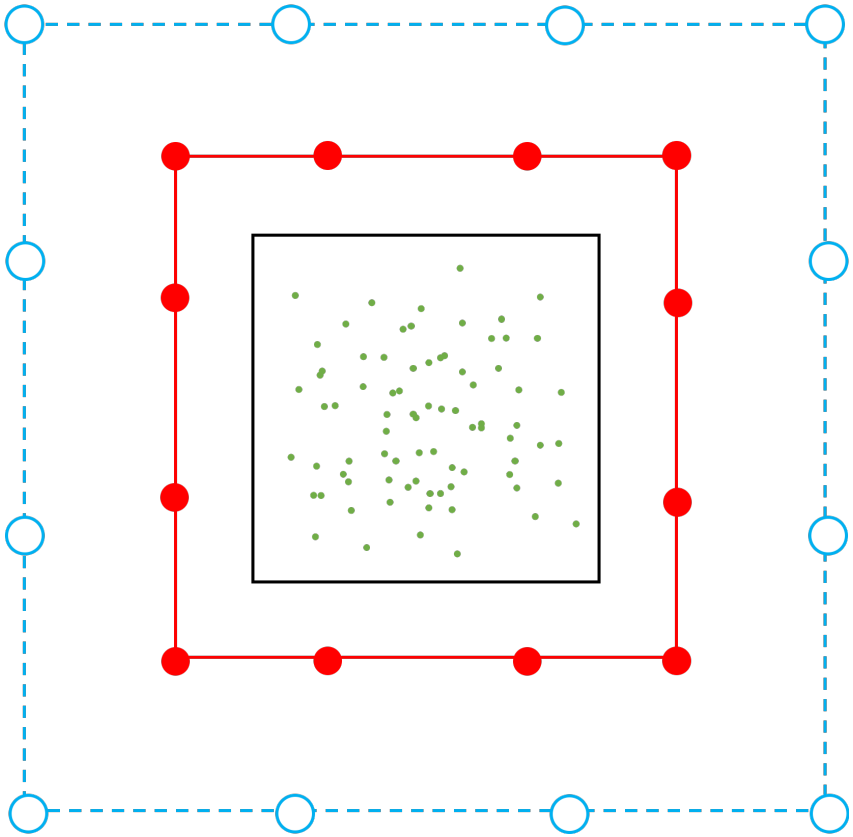
Analytic FMM - Implementation Issues

- Representing problem with efficient data structures: Quad/Octrees
- Computing and storing new expansions for each kernel, may require new software implementations

Kernel Independent FMM

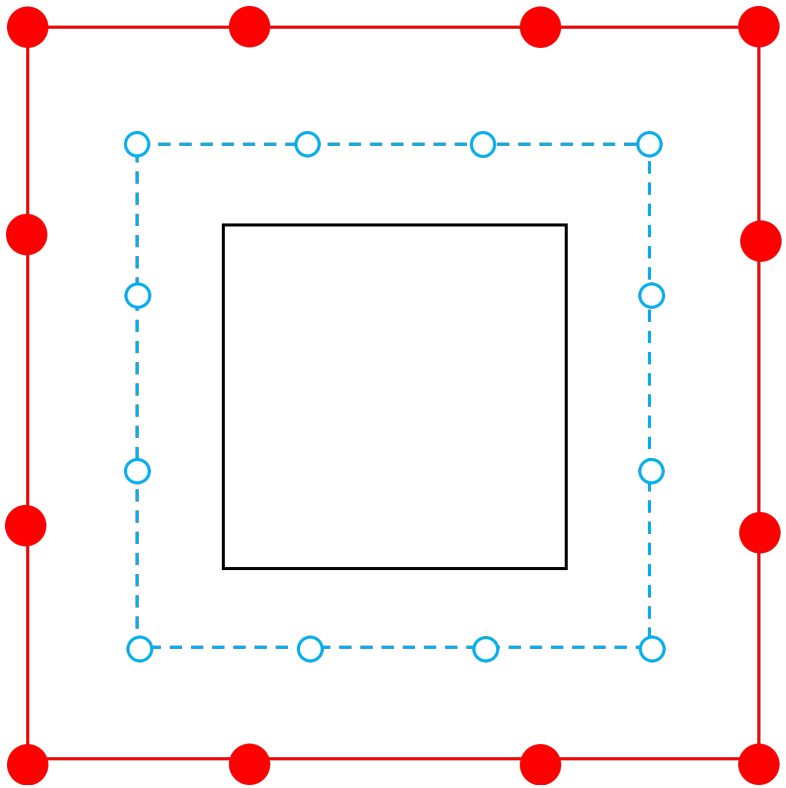
- KIFMM only requires kernel evaluations
- Works by matching potential generated by particles with that generated by an equivalent density in the far field

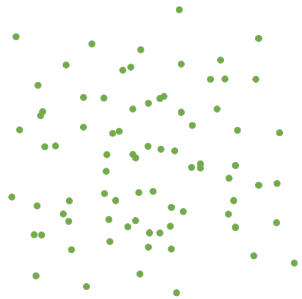
Kernel Independent FMM - Least Squares Problem



Adapted from [2]

Kernel Independent FMM - Least Squares Problem





Adapted from [2]

Kernel Independent FMM - Least Squares Problem

- Check Surface $x^{B,u}$
- Equivalent Surface $y^{B,u}$
- Equivalent Density $\phi^{B,u}$
- Check Potential $q^{B,u}$
- Indices of source points I_s^B
- Source densities ϕ_i

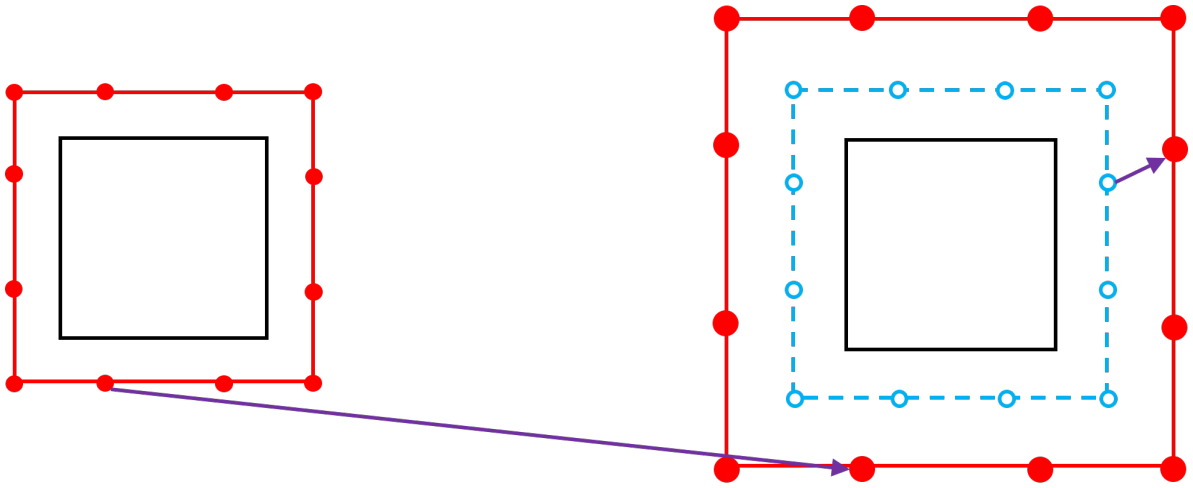
$$\int_{\mathrm{y}^{B,u}} K(\mathrm{x},\mathrm{y})\phi^{B,u} d\mathrm{y} = \sum_{i \in I_s^B} K(\mathrm{x},\mathrm{y})\phi_i = q^{B,u} \text{ for any } \mathrm{x} \in \mathrm{x}^{B,u} \quad (3)$$

Kernel Independent FMM - Least Squares Problem

$$K_A \phi^A = K_B \phi^B \tag{4}$$

$$\phi^A = (\alpha I + K_A^* K_A)^{-1} K_B \phi^B \tag{5}$$

Kernel Independent FMM - Least Squares Problem, M2L



Kernel Independent FMM - Least Squares Problem, M2L

- Check Surface $x^{B,u}$
- Downward Equivalent Surface $y^{B,d}$
- Upward Equivalent Surface $y^{A,u}$
- Downward Equivalent Density $\phi^{B,d}$
- Upward Equivalent Density $\phi^{A,u}$

$$\int_{\mathrm{y}^{A,u}} K(\mathrm{x},\mathrm{y})\phi^{A,u} d\mathrm{y} = \int_{\mathrm{y}^{B,d}} K(\mathrm{x},\mathrm{y})\phi^{B,d} d\mathrm{y}, \quad \text{for any } \mathrm{x} \in \mathrm{x}^{B,d} \qquad (6)$$

Kernel Independent FMM - Implementation Issues

- No need for new software implementation for large class of compatible Kernels
- Can built singly, extensible, and optimisable software implementation

Table of Contents

Motivation

- Python has emerged as a standard in scientific and data intensive computing
- Desire a high quality software implementation which is also highly performant and easily portable
- Tradeoff performance of compiled languages for engineering ease, and portability

Goals

- Create a performant 3D Python implementation of the KIFMM
- Write software in an extensible and well tested way
- Take advantage of distributed and parallel computing concepts as much as possible

Outcomes - Vectorised Data Structures

Outcomes - JIT Compilation

Outcomes - Multiprocessing

Outcomes - Low-Rank SVD Compression

Outcomes - Extensible Software Design

Table of Contents

Modern Architectures

Randomised SVD Compression

References I

[1] L Greengard and V Rokhlin.
A fast algorithm for particle simulations.
Journal of Computational Physics, 73(2):325 – 348, 1987.

[2] Denis Zorin Lexing Ying, George Biros.
A kernel-independent adaptive fast multipole algorithm in two and three dimensions.
196(2):591–626, 2004.