gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

An Introduction to Fast Multipole Methods

Ramani Duraiswami
Institute for Advanced Computer Studies
University of Maryland, College Park
http://www.umiacs.umd.edu/~ramani

Joint work with Nail A. Gumerov

Fast Multipole Methods

- Computational simulation is becoming an accepted paradigm for scientific discovery.
  - Many simulations involve several million variables
- Most large problems boil down to solution of linear system or performing a matrix-vector product
- Regular product requires $O(N^2)$ time and $O(N^2)$ memory
- The FMM is a way to
  - accelerate the products of particular dense matrices with vectors
  - Do this using $O(N)$ memory
- FMM achieves product in $O(N)$ or $O(N \log N)$ time and memory
- Combined with iterative solution methods, can allow solution of problems hitherto unsolvable

Matrix vector product

$s_1 = m_{11} x_1 + m_{12} x_2 + \ldots + m_{1d} x_d$
$s_2 = m_{21} x_1 + m_{22} x_2 + \ldots + m_{2d} x_d$
…
$s_n = m_{n1} x_1 + m_{n2} x_2 + \ldots + m_{nd} x_d$

- Matrix vector product is identical to a sum

  $s_i = \sum_{j=1}^{d} m_{ij} x_j$
- So algorithm for fast matrix vector products is also a fast summation algorithm

- d products and sums per line
- N lines
- Total Nd products and Nd sums to calculate N entries
- Memory needed is NM entries

Linear Systems

- Solve a system of equations

  $$Mx=s$$
- M is a $N \times N$ matrix, x is a N vector, s is a N vector
- Direct solution (Gauss elimination, LU Decomposition, SVD, …) all need $O(N^3)$ operations
- Iterative methods typically converge in k steps with each step needing a matrix vector multiply $O(N^2)$
  - if properly designed, $k << N$
- A fast matrix vector multiplication algorithm requiring $O(N \log N)$ operations will speed all these algorithms

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Is this important?

- Argument:
  - Moore's law: Processor speed doubles every 18 months
  - If we wait long enough the computer will get fast enough and let my inefficient algorithm tackle the problem
- Is this true?
  - Yes for algorithms with same asymptotic complexity
  - No!! For algorithms with different asymptotic complexity
- For a million variables, we would need about 16 generations of Moore's law before a $O(N^2)$ algorithm is comparable with a $O(N)$ algorithm
- Similarly, clever problem formulation can also achieve large savings.

Memory complexity

- Sometimes we are not able to fit a problem in available memory
  - Don't care how long solution takes, just if we can solve it
- To store a $N \times N$ matrix we need N^2 locations
  - 1 GB RAM = 1024^3 =1,073,741,824 bytes
  - => largest N is 32,768
- "Out of core" algorithms copy partial results to disk, and keep only necessary part of the matrix in memory
  - Extremely slow
- FMM allows reduction of memory complexity as well
  - *Elements of the matrix required for the product can be generated as needed*
  - Can solve much larger problems (e.g., 10^7 variables on a PC)

The need for fast algorithms

- Grand challenge problems in large numbers of variables
- Simulation of physical systems
  - Electromagnetics of complex systems
  - Stellar clusters
  - Protein folding
  - Acoustics
  - Turbulence
- Learning theory
  - Kernel methods
  - Support Vector Machines
- Graphics and Vision
  - Light scattering …

- General problems in these areas can be posed in terms of millions (10^6) or billions (10^9) of variables
- Recall Avogadro's number ($6.022\,141\,99 \times 10^{23}$ molecules/mole
- Job of modeling is to find symmetries and representations that reduce the size of the problem
- Even after state of art modeling, problem size may be large

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Dense and Sparse matrices

- Operation estimates are for **dense matrices**.
  - ❑Majority of elements of the matrix are ***non-zero***
- However in many applications matrices are ***sparse***
- A sparse matrix (or vector, or array) is one in which most of the elements are zero.
  - ❑If storage space is more important than access speed, it may be preferable to store a sparse matrix as a list of (index, value) pairs.
  - ❑For a given sparsity structure it may be possible to define a fast matrix-vector product/linear system algorithm

- Can compute

$$\begin{bmatrix} a_1 & 0 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 & 0 \\ 0 & 0 & a_3 & 0 & 0 \\ 0 & 0 & 0 & a_4 & 0 \\ 0 & 0 & 0 & 0 & a_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} a_1 x_1 \\ a_2 x_2 \\ a_3 x_3 \\ a_4 x_4 \\ a_5 x_5 \end{bmatrix}$$

In 5 operations instead of 25 operations

- Sparse matrices are not our concern here

Structured matrices

- Fast algorithms have been found for many dense matrices
- Typically the matrices have some *"structure"*
- Definition:
  - ❑A dense matrix of order $N \times N$ is called structured if its entries depend on only $O(N)$ parameters.
- Most famous example – the fast Fourier transform

Fourier Matrices

A Fourier matrix of order n is defined as the following

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix},$$

where

$$\omega_n = e^{-\frac{2\pi i}{n}},$$

is an nth root of unity.

FFT presented by Cooley and Tukey in 1965, but invented several times, including by Gauss (1809) and Danielson & Lanczos (1948)

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

FFT and IFFT

The *discrete Fourier transform* of a vector x is the product $F_n x$.
The *inverse discrete Fourier transform* of a vector x is the product $F_n^* x$.

Both products can be done efficiently using the fast Fourier transform (FFT) and the inverse fast Fourier transform (IFFT) in $O(n \log n)$ time.

The FFT has revolutionized many applications by reducing the complexity by a factor of almost *n*

Can relate many other matrices to the Fourier Matrix

Circulant Matrices

$$C_n = C(x_1, ..., x_n) = \begin{bmatrix} x_1 & x_n & x_{n-1} & \cdots & x_2 \\ x_2 & x_1 & x_n & \cdots & x_3 \\ x_3 & x_2 & x_1 & \cdots & x_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_n & x_{n-1} & x_{n-2} & \cdots & x_1 \end{bmatrix}$$

Toeplitz Matrices

$$T_n = T(x_{-n+1}, \cdots, x_0, ..., x_{n-1}) = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ x_{-1} & x_0 & x_1 & \cdots & x_{n-2} \\ x_{-2} & x_{-1} & x_0 & \cdots & x_{n-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{-n+1} & x_{-n+2} & x_{-n+3} & \cdots & x_0 \end{bmatrix}$$

Hankel Matrices

$$H_n = H(x_{-n+1}, \cdots, x_0, \cdots, x_{n-1}) = \begin{bmatrix} x_{-n+1} & x_{-n+2} & x_{-n+3} & \cdots & x_0 \\ x_{-n+2} & x_{-n+3} & x_{-n+4} & \cdots & x_1 \\ x_{-n+3} & x_{-n+4} & x_{-n+5} & \cdots & x_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0 & x_1 & x_2 & \cdots & x_{n-1} \end{bmatrix}$$

Vandermonde Matrices

$$V = V(x_0, x_1, ..., x_n) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ x_0^{n-1} & x_1^{n-1} & \cdots & x_{n-1}^{n-1} \end{bmatrix}$$

Structured Matrices

- (usually) these matrices can be diagonalized by the Fourier matrix
- Product of diagonal matrix and vector requires O(N) operations
- So complexity is the cost of FFT (O ($N \log N$)) + product (O(N))
- Order notation
  - Only keep leading order term (asymptotically important)
  - So complexity of the above is O ($N \log N$)
- Structured Matrix algorithms are "brittle"
  - FFT requires uniform sampling
  - Slight departure from uniformity breaks factorization

Fast Multipole Methods (FMM)

- Introduced by Rokhlin & Greengard in 1987
- Called one of the 10 most significant advances in computing of the 20th century
- Speeds up matrix-vector products (sums) of a particular type

$$s(x_j) = \sum_{i=1}^{N} \alpha_i \phi(x_j - x_i), \quad \{s_j\} = [\Phi_{ji}]\{\alpha_i\}.$$

- Above sum requires $O(MN)$ operations.
- For a given precision ε the FMM achieves the evaluation in $O(M+N)$ operations.
- Edelman: *"FMM is all about adding functions"*
  - *Talk on Tuesday, next week*

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Is the FMM a structured matrix algorithm?

- FFT and other algorithms work on structured matrices
- What about FMM ?
- Speeds up matrix-vector products (sums) of a particular type

$$s(\mathbf{y}_j) = \sum_{i=1}^{N} a_i \phi(\mathbf{x}_i, \mathbf{y}_j)$$

$$\{s_j\} = [\Phi_{ji}]\{\alpha_i\}.$$

$$\mathbf{s} = \mathbf{\Phi a}$$

- Above sum also depends on $O(N)$ parameters $\{x_i\}$, $\{y_j\}$, ϕ
- FMM can be thought of as working on "loosely" structured matrices

- Can accelerate matrix vector products
  - ❑ Convert $O(N^2)$ to $O(N \log N)$
- However, can also accelerate linear system solution
  - ❑ Convert $O(N^3)$ to $O(kN \log N)$
  - ❑ For some iterative schemes can guarantee $k \leq N$
  - ❑ In general, goal of research in iterative methods is to reduce value of k
  - ❑ Well designed iterative methods can converge in very few steps
  - ❑ Active research area: design iterative methods for the FMM

A very simple algorithm

- Not FMM, but has some key ideas
- Consider

$$S(x_i)=\sum_{j=1}^{N} \alpha_j (x_i - y_j)^2 \qquad i=1, \dots ,M$$

- Naïve way to evaluate the sum will require MN operations
- Instead can write the sum as

$$S(x_i)=(\sum_{j=1}^{N} \alpha_j)x_i^2 + (\sum_{j=1}^{N} \alpha_j y_j^2) -2x_i(\sum_{j=1}^{N} \alpha_j y_j)$$

  - ❑ Can evaluate each bracketed sum over j and evaluate an expression of the type

$$S(x_i)=\beta x_i^2 + \gamma -2x_i\delta$$

  - ❑ Requires O(M+N) operations
- Key idea – use of analytical manipulation of series to achieve faster summation
- May not always be possible to simply factorize matrix entries

Approximate evaluation

- FMM introduces another key idea or "philosophy"
  - ❑ In scientific computing we almost never seek exact answers
  - ❑ At best, "exact" means to "machine precision"
- So instead of solving the problem we can solve a "nearby" problem that gives "almost" the same answer
  - ❑ If this "nearby" problem is much easier to solve, and we can bound the error analytically we are done.
- In the case of the FMM
  - ❑ Express functions in some appropriate functional space with a given basis
  - ❑ Manipulate series to achieve approximate evaluation
  - ❑ Use analytical expression to bound the error
- FFT is exact … FMM can be arbitrarily accurate

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Approximation Algorithms

- Computer science approximation algorithms
  - ❑ Approximation algorithms are usually directed at reducing complexity of exponential algorithms by performing approximate computations
  - ❑ Here the goal is to reduce polynomial complexity to linear order
  - ❑ Connections between FMM and CS approximation algorithms are not much explored

Tree Codes

- Idea of approximately evaluating matrix vector products preceded FMM
- Tree codes (Barnes and Hut, 1986)
- Divides domain into regions and use approximate representations
- Key difference: lack error bounds, and automatic ways of adjusting representations
- Perceived to be easier to program

Complexity

- The most common complexities are
  - ❑ O(1) - not proportional to any variable number, i.e. a fixed/constant amount of time
  - ❑ O(N) - proportional to the size of N (this includes a loop to N and loops to constant multiples of N such as 0.5N, 2N, 2000N - no matter what that is, if you double N you expect (on average) the program to take twice as long)
  - ❑ O(N^2) - proportional to N squared (you double N, you expect it to take four times longer - usually two nested loops both dependent on N).
  - ❑ O(log N) - this is tricker to show - usually the result of binary splitting.
  - ❑ O(N log N) this is usually caused by doing log N splits but also doing N amount of work at each "layer" of splitting.
  - ❑ Exponential $O(a^N)$: grows faster than any power of N

Some FMM algorithms

- Molecular and stellar dynamics
  - ❑ Computation of force fields and dynamics
- Interpolation with Radial Basis Functions
- Solution of acoustical scattering problems
  - ❑ Helmholtz Equation
- Electromagnetic Wave scattering
  - ❑ Maxwell's equations
- Fluid Mechanics: Potential flow, vortex flow
  - ❑ Laplace/Poisson equations
- Fast nonuniform Fourier transform

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Integral Equation

- FMM is often used in integral equations
- What is an integral equation?

$$\int k(x,y)u(x)dx + au(y) = f(y)$$

$$\int k(x,y)u(x)dx = f(y)$$

- Function *k(x,y)* is called the kernel
- Integral equations are typically solved by "quadrature"
  - Quadrature is the process of approximately evaluating an integral
- If we can write

$$\int k(x,y)u(x)dx = \sum_{j=1}^{N} k(x_j,y)u(x_j)w_j$$

FMM-able Matrices

$$\mathbf{v} = \mathbf{\Phi u},$$

$$\mathbf{\Phi} = \begin{pmatrix} \Phi(\mathbf{y}_1,\mathbf{x}_1) & \Phi(\mathbf{y}_1,\mathbf{x}_2) & \dots & \Phi(\mathbf{y}_1,\mathbf{x}_N) \\ \Phi(\mathbf{y}_2,\mathbf{x}_1) & \Phi(\mathbf{y}_2,\mathbf{x}_2) & \dots & \Phi(\mathbf{y}_2,\mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ \Phi(\mathbf{y}_M,\mathbf{x}_1) & \Phi(\mathbf{y}_M,\mathbf{x}_2) & \dots & \Phi(\mathbf{y}_M,\mathbf{x}_N) \end{pmatrix}.$$

$$X = \{\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N\}, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad i = 1,\dots,N,$$
$$Y = \{\mathbf{y}_1,\mathbf{y}_2,\dots,\mathbf{y}_M\}, \quad \mathbf{y}_j \in \mathbb{R}^d, \quad j = 1,\dots,M.$$

$$v_j = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}_j,\mathbf{x}_i), \quad j = 1,\dots,M.$$

Factorization

Degenerate Kernel:

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j).$$

$$v_j = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{i=1}^{N} u_i \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j)$$
$$= \sum_{m=0}^{p-1}\left[\sum_{i=1}^{N} u_i A_m(\mathbf{x}_i)\right]F_m(\mathbf{y}_j) = \sum_{m=0}^{p-1} B_m F_m(\mathbf{y}_j).$$

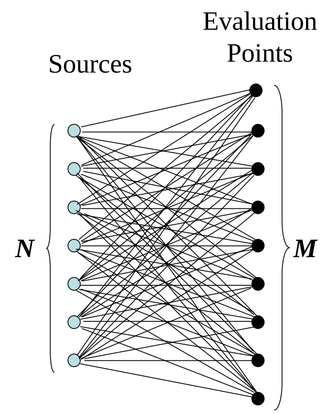
O(*pN*) operations: $B_m = \sum_{i=1}^{N} u_i A_m(\mathbf{x}_i), \quad m = 0,\dots,p-1,$

O(*pM*) operations: $v_j = \sum_{m=0}^{p-1} B_m F_m(\mathbf{y}_j), \quad j = 1,\dots,M.$

Total Complexity: *O*(*p*(*N*+*M*))

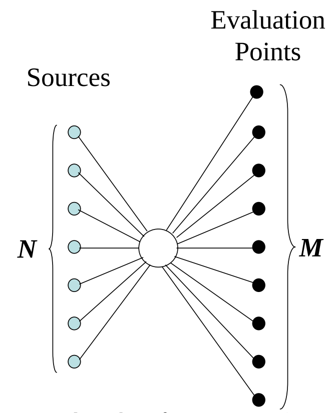
"Middleman" Algorithm

Standard algorithm



Total number of operations: ***O(NM)***

Middleman algorithm



Total number of operations: ***O(N+M)***

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Factorization

Non-Degenerate Kernel: Truncation Number

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j) + Error(p,\mathbf{x}_i,\mathbf{y}_j).$$

$$v_j = \sum_{i=1}^{N} u_i\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{i=1}^{N} u_i \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j) + \sum_{i=1}^{N} u_i Error(p,\mathbf{x}_i,\mathbf{y}_j)$$

$$= \sum_{m=0}^{p-1} B_m F_m(\mathbf{y}_j) + Error_j(p,N), \quad j = 1,\dots,M.$$

Error Bound: $|Error_j(p,N)| < N\max_i|u_i|\max_i|Error(p,\mathbf{x}_i,\mathbf{y}_j)|.$

Middleman Algorithm Applicability: $p \ll \min(M,N),$ $|Error_j(p,N)| < \epsilon.$

Factorization Problem:

•Usually there is no factorization available that provides a uniform approximation of the kernel in the entire computational domain.
•So we have to construct a patchwork-quilt of overlapping approximations, and manage this.

•Need representations of functions that allow this
•Need data structures for the management

Five Key Stones of FMM

- Representation and Factorization
- Error Bounds and Truncation
- Translation
- Space Partitioning
- Data Structures

Fast Multipole Methods

- Middleman (separation of variables)
  - ❑ No space partitioning
- Single Level Methods
  - ❑ Simple space partitioning (usually boxes)
- Multilevel FMM (MLFMM)
  - ❑ Multiple levels of space partitioning (usually hierarchical boxes)
- Adaptive MLFMM
  - ❑ Data dependent space partitioning

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Examples of Matrices

- Green's functions of Laplace and Helmholtz equations

$$\Phi(\mathbf{y},\mathbf{x}) = \frac{1}{4\pi|\mathbf{y}-\mathbf{x}|},$$

$$\Phi(\mathbf{y},\mathbf{x}) = \frac{\exp\{ik|\mathbf{y}-\mathbf{x}|\}}{4\pi|\mathbf{y}-\mathbf{x}|}.$$

- Potential velocity field of a source located at $\mathbf{x}_i$

$$\boldsymbol{\Phi}(\mathbf{y},\mathbf{x}_i) = \mathbf{V}(\mathbf{y},\mathbf{x}_i) = \frac{1}{4\pi}\nabla_{\mathbf{y}}\frac{1}{|\mathbf{y}-\mathbf{x}_i|}.$$

- Normal derivative on the surface

$$\Phi(\mathbf{y},\mathbf{x}) = \frac{\partial}{\partial n(\mathbf{x})}\frac{1}{4\pi|\mathbf{y}-\mathbf{x}|} = \mathbf{n}(\mathbf{x})\cdot\nabla_{\mathbf{x}}\frac{1}{4\pi|\mathbf{y}-\mathbf{x}|}.$$

- Vorticity (vortex element is located at $\mathbf{x}_i$)

$$\Phi(\mathbf{y},\mathbf{x}_i) = \nabla_{\mathbf{y}}\times\mathbf{V}(\mathbf{y},\mathbf{x}_i).$$

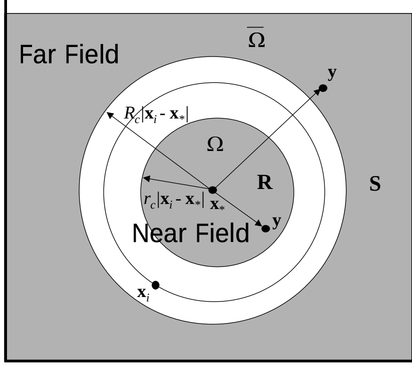
Iterative Methods

- To solve linear systems of equations;
- Simple iteration methods;
- Conjugate gradient or similar methods;
- We use Krylov subspace methods:
  - ❑Parameters of the method;
  - ❑Preconditioners;
  - ❑Research is ongoing.
- Efficiency critically depends on efficiency of the matrix-vector multiplication.

Far and Near Field Expansions

Far Field: $\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{p-1} C_m(\mathbf{x}_i,\mathbf{x}_*)S_m(\mathbf{y}_j-\mathbf{x}_*) + Error.$

Near Field: $\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{p-1} D_m(\mathbf{x}_i,\mathbf{x}_*)R_m(\mathbf{y}_j-\mathbf{x}_*) + Error.$



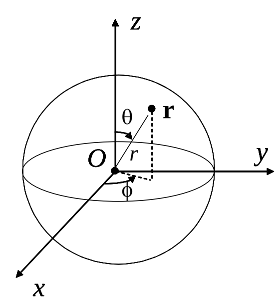
S: "Singular"
(also "Multipole",
"Outer"
"Far Field"),

R: "Regular"
(also "Local",
"Inner"
"Near Field")

Example of Multipole and Local expansions (3D Laplace)

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{n=0}^{p-1}\sum_{m=-n}^{n} C_n^m S_n^m(\mathbf{y}_j-\mathbf{x}_*) + Error(p),$$

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{n=0}^{p-1}\sum_{m=-n}^{n} D_n^m R_n^m(\mathbf{y}_j-\mathbf{x}_*) + Error(p),$$



$$S_n^m(\mathbf{r}) = \frac{(-1)^n i^{|m|}}{\alpha_n^m}\sqrt{\frac{4\pi}{2n+1}}\frac{1}{r^{n+1}}Y_n^m(\theta,\varphi),$$

$$R_n^m(\mathbf{r}) = i^{-|m|}\alpha_n^m\sqrt{\frac{4\pi}{2n+1}}r^n Y_n^m(\theta,\varphi),$$

$$\alpha_n^m = \alpha_n^{-m} = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}.$$

Spherical Coordinates:

$\mathbf{r} = r(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$

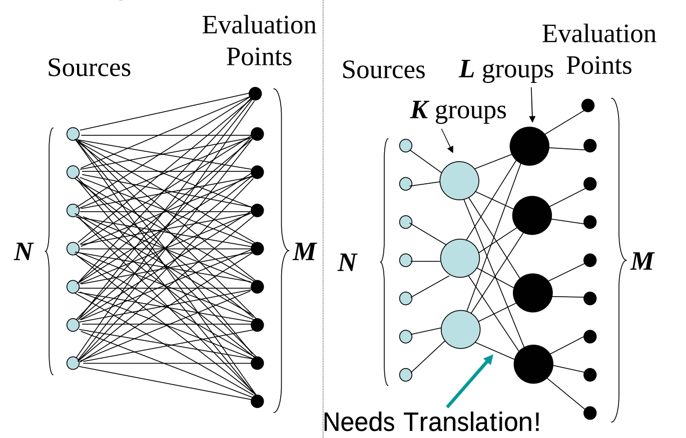
Spherical Harmonics:

$$Y_n^m(\theta,\varphi) = (-1)^m\sqrt{\frac{2n+1}{4\pi}\frac{(n-|m|)!}{(n+|m|)!}}P_n^{|m|}(\cos\theta)e^{im\varphi}$$

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Idea of a Single Level FMM

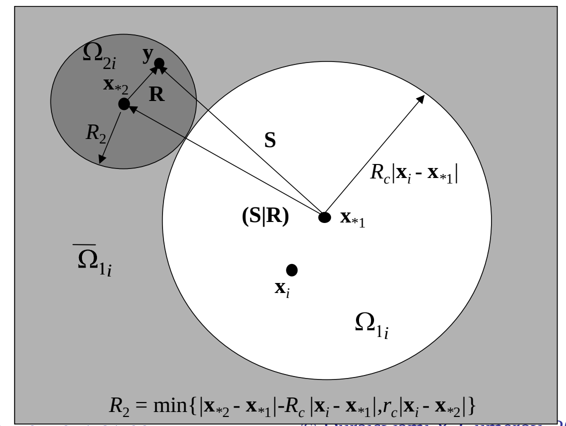


Total number of operations: ***O(NM)***

Total number of operations: ***O(N+M+KL)***

Multipole-to-Local S|R-translation

Also "Far-to-Local", "Outer-to-Inner", "Multipole-to-Local"



S|R-translation Operator

$$\Phi(\mathbf{y}) = \sum_{m=0}^{p-1} C_m S_m(\mathbf{y} - \mathbf{x}_{*1}) + Error.$$

$$\Phi(\mathbf{y}) = \sum_{m=0}^{p-1} D_m R_m(\mathbf{y} - \mathbf{x}_{*2}) + Error.$$

S|R-Translation Coefficients

$$S_n(\mathbf{y} - \mathbf{x}_{*1}) = \sum_{m=0}^{p-1} (S|R)_{mn}(\mathbf{x}_{*2} - \mathbf{x}_{*1}) R_m\left(\mathbf{y}_j - \mathbf{x}_{*2}\right) + Error.$$

S|R-Translation Matrix

$$D_m(\mathbf{y} - \mathbf{x}_{*1}) = \sum_{m=0}^{p-1} (S|R)_{mn}(\mathbf{x}_{*2} - \mathbf{x}_{*1}) C_n\left(\mathbf{y}_j - \mathbf{x}_{*2}\right) + Error.$$

S|R-translation Operators for 3D Laplace and Helmholtz equations

$$\Phi(\mathbf{y}) = \sum_{n=0}^{p-1} \sum_{m=-n}^{n} C_n^m S_n^m(\mathbf{y} - \mathbf{x}_{*1}) + Error.$$

$$\Phi(\mathbf{y}) = \sum_{n=0}^{p-1} \sum_{m=-n}^{n} D_n^m R_n^m(\mathbf{y} - \mathbf{x}_{*2}) + Error.$$

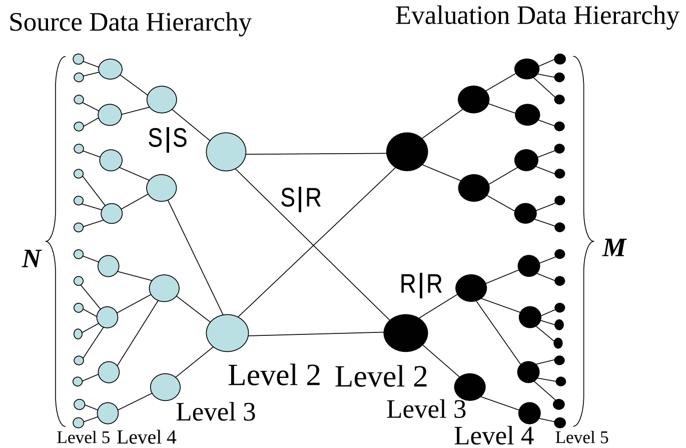
$$S_n^m(\mathbf{y} - \mathbf{x}_{*1}) = \sum_{n'=0}^{p-1} \sum_{m'=-n'}^{n'} (S|R)_{n'n}^{m'm}(\mathbf{x}_{*2} - \mathbf{x}_{*1}) R_{n'}^{m'}\left(\mathbf{y}_j - \mathbf{x}_{*2}\right) + Error.$$

$$D_n^m(\mathbf{y} - \mathbf{x}_{*1}) = \sum_{n'=0}^{p-1} \sum_{m'=-n'}^{n'} (S|R)_{nn'}^{mm'}(\mathbf{x}_{*2} - \mathbf{x}_{*1}) C_{n'}^{m'}\left(\mathbf{y}_j - \mathbf{x}_{*2}\right) + Error.$$

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Idea of Multilevel FMM



Complexity of Translation

- For 3D Laplace and Helmholtz series have p^2 terms;
- Translation matrices have p^4 elements;
- Translation performed by direct matrix-vector multiplication has complexity $O(p^4)$;
- Can be reduced to $O(p^3)$;
- Can be reduced to $O(p^2 \log^2 p)$;
- Can be reduced to $O(p^2)$ (?).

Week 2: Representations

- **Gregory Beylkin** (University of Colorado) "Separated Representations and Fast Adaptive Algorithms in Multiple Dimensions"
- **Alan Edelman** (MIT) "Fast Multipole: It's All About Adding Functions in Finite Precision"
- **Vladimir Rokhlin** (Yale University) "Fast Multipole Methods in Oscillatory Environments: Overview and Current State of Implementation"
- **Ramani Duraiswami** (University of Maryland) "An Improved Fast Gauss Transform and Applications"
- **Eric Michielssen** (University of Illinois at Urbana-Champaign) "Plane Wave Time Domain Accelerated Integral Equation Solvers"

Week 2: Data Structures

- **David Mount** (University of Maryland) "Data Structures for Approximate Proximity and Range Searching"
- **Alexander Gray** (Carnegie Mellon University) "New Lightweight N-body Algorithms"
- **Ramani Duraiswami** (University of Maryland) "An Improved Fast Gauss Transform and Applications"

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Week 2: Applications

- **Nail Gumerov** (University of Maryland) "Computation of 3D Scattering from Clusters of Spheres using the Fast Multipole Method"
- **Weng Chew** (University of Illinois at Urbana-Champaign) "Review of Some Fast Algorithms for Electromagnetic Scattering"
- **Leslie Greengard** (Courant Institute, NYU) "FMM Libraries for Computational Electromagnetics"
- **Qing Liu** (Duke University) "NUFFT, Discontinuous Fast Fourier Transform, and Some Applications"
- **Eric Michielssen** (University of Illinois at Urbana-Champaign) "Plane Wave Time Domain Accelerated Integral Equation Solvers"
- **Gregory Rodin** (University of Texas, Austin) "Periodic Conduction Problems: Fast Multipole Method and Convergence of Integral Equations and Lattice Sums"
- **Stephen Wandzura** (Hughes Research Laboratories) "Fast Methods for Fast Computers"
- **Toru Takahashi** (Institue of Physical and Chemical Research (RIKEN), Japan) "Fast Computing of Boundary Integral Equation Method by a Special-purpose Computer"
- **Ramani Duraiswami** (University of Maryland) "An Improved Fast Gauss Transform and Applications"

Tree Codes:

- **Atsushi Kawai** (Saitama Institute of Technology) "Fast Algorithms on GRAPE Special-Purpose Computers"
- **Walter Dehnen** (University of Leicester) "falcON: A Cartesian FMM for the Low-Accuracy Regime"
- **Robert Krasny** (University of Michigan) "A Treecode Algorithm for Regularized Particle Interactions“
- **Derek Richardson** (University of Maryland) "pkdgrav: A Parallel k-D Tree Gravity Solver for N-Body Problems"

Key Ideas of the FMM

Nail Gumerov &
Ramani Duraiswami
UMIACS
[gumerov][ramani]@umiacs.umd.edu

Content

- Summation Problems
- Factorization (Middleman Method)
- Space Partitioning (Modified Middleman Method)
- Translations (Single Level FMM)
- Hierarchical Space Partitioning (Multilevel FMM)

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Summation Problems

Matrix-Vector Multiplication

Compute matrix vector product

$$\mathbf{v} = \Phi\mathbf{u}$$

or

$$v_j = \sum_{i=1}^{N} \Phi_{ji} u_i, \quad j = 1,\ldots,M,$$

where

$$\Phi_{ji} = \Phi(\mathbf{y}_j, \mathbf{x}_i), \quad j = 1,\ldots,M, \quad i = 1,\ldots,N,$$

or

$$\Phi = \begin{pmatrix} \Phi_{11} & \Phi_{12} & \ldots & \Phi_{1N} \\ \Phi_{21} & \Phi_{22} & \ldots & \Phi_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ \Phi_{M1} & \Phi_{M2} & \ldots & \Phi_{MN} \end{pmatrix} = \begin{pmatrix} \Phi(\mathbf{y}_1,\mathbf{x}_1) & \Phi(\mathbf{y}_1,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_1,\mathbf{x}_N) \\ \Phi(\mathbf{y}_2,\mathbf{x}_1) & \Phi(\mathbf{y}_2,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_2,\mathbf{x}_N) \\ \ldots & \ldots & \ldots & \ldots \\ \Phi(\mathbf{y}_M,\mathbf{x}_1) & \Phi(\mathbf{y}_M,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_M,\mathbf{x}_N) \end{pmatrix}.$$

Generally we have two sets of points in d-dimensions:

$$\text{Sources}: \mathbb{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad i = 1,\ldots,N,$$
$$\text{Receivers}: \mathbb{Y} = \{\mathbf{y}_1,\ldots,\mathbf{y}_M\}, \quad \mathbf{y}_j \in \mathbb{R}^d, \quad j = 1,\ldots,M,$$

The receivers also can be called "targets" or "evaluation points".

Why **R**^{d} ?

- d = 1
  - Scalar functions, interpolation, etc.
- d = 2,3
  - Physical problems in 2 and 3 dimensional space
- d = 4
  - 3D Space + time, 3D grayscale images
- d = 5
  - Color 2D images, Motion of 3D grayscale images
- d = 6
  - Color 3D images
- d = 7
  - Motion of 3D color images
- d = arbitrary
  - d-parametric spaces, statistics, database search procedures

Fields (Potentials)

Field (Potential) of a single (ith) unit source

$$v(\mathbf{y}) = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}, \mathbf{x}_i), \quad \mathbf{y} \in \mathbb{R}^d,$$

$$v_j = v(\mathbf{y}_j), \quad j = 1,\ldots,M.$$

Field (Potential) of the set of sources of intensities $\{u_i\}$

Fields are continuous!
(Almost everywhere)

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Examples of Fields

- There can be vector or scalar fields (we focus mostly on scalar fields)
- Fields can be *regular* or *singular*

Scalar Fields:

Gravity

(singular at $\mathbf{y} = \mathbf{x}_i$)

$$\Phi(\mathbf{y},\mathbf{x}_i) = \frac{1}{|\mathbf{y}-\mathbf{x}_i|}$$

Monochromatic Wave (k is the wavenumber)

(singular at $\mathbf{y} = \mathbf{x}_i$)

$$\Phi(\mathbf{y},\mathbf{x}_i) = \frac{\exp\{ik|\mathbf{y}-\mathbf{x}_i|\}}{|\mathbf{y}-\mathbf{x}_i|}$$

Gaussian

(regular everywhere)

$$\Phi(\mathbf{y},\mathbf{x}_i) = \exp\{-|\mathbf{y}-\mathbf{x}_i|^2/\sigma\}$$

Vector Field:

3D Velocity field:

(singular at $\mathbf{y} = \mathbf{x}_i$)

$$\boldsymbol{\Phi}(\mathbf{y},\mathbf{x}_i) = \nabla_y \frac{1}{|\mathbf{y}-\mathbf{x}_i|} = \mathbf{i}_1\frac{\partial}{\partial y_1}\frac{1}{|\mathbf{y}-\mathbf{x}_i|} + \mathbf{i}_2\frac{\partial}{\partial y_2}\frac{1}{|\mathbf{y}-\mathbf{x}_i|} + \mathbf{i}_3\frac{\partial}{\partial y_3}\frac{1}{|\mathbf{y}-\mathbf{x}_i|},$$

$$\mathbf{y} = (y_1, y_2, y_3) \in \mathbb{R}^3.$$

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Straightforward Computational Complexity:

$O(MN)$ Error: 0 (“machine” precision)

The Fast Multipole Methods look for computation of the same problem with complexity $o(MN)$ and error < prescribed error.

In the case when the error of the FMM does not exceed the machine precision error (for given number of bits) there is no difference between the “exact” and “approximate” solution.

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Factorization
“Middleman Method”

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Global Factorization

$\forall \mathbf{x}_i, \mathbf{y}_j \in \Omega \subset \mathbb{R}^d$:

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{\infty} a_m(\mathbf{x}_i - \mathbf{x}_*) f_m(\mathbf{y}_j - \mathbf{x}_*) = \sum_{m=0}^{p-1} a_m(\mathbf{x}_i - \mathbf{x}_*) f_m(\mathbf{y}_j - \mathbf{x}_*) + Error(p, \mathbf{x}_i, \mathbf{y}_j)$$

Expansion center

Truncation number

Expansion coefficients

Basis functions

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Factorization Trick

$$v_j = \sum_{i=1}^{N} \Phi(\mathbf{y}_j, \mathbf{x}_i) u_i$$
$$= \sum_{i=1}^{N} \left[\sum_{m=0}^{p-1} a_m(\mathbf{x}_i - \mathbf{x}_*) f_m(\mathbf{y}_j - \mathbf{x}_*) + Error(p, \mathbf{x}_i, \mathbf{y}_j) \right] u_i$$
$$= \sum_{m=0}^{p-1} f_m(\mathbf{y}_j - \mathbf{x}_*) \sum_{i=1}^{N} a_m(\mathbf{x}_i - \mathbf{x}_*) u_i + \sum_{i=1}^{N} Error(p, \mathbf{x}_i, \mathbf{y}_j) u_i$$
$$= \sum_{m=0}^{p-1} c_m f_m(\mathbf{y}_j - \mathbf{x}_*) + Error(N, p),$$

where

$$c_m = \sum_{i=1}^{N} a_m(\mathbf{x}_i - \mathbf{x}_*) u_i.$$

Reduction of Complexity

Straightforward (nested loops):

```
for j = 1,...,M
    v_j = 0;
    for i = 1,...,N
        v_j = v_j + Φ(y_j, x_i) u_i;
    end;
end;
```

Complexity: $O(MN)$

If $p << \min(M,N)$ then complexity reduces!

Factroized:

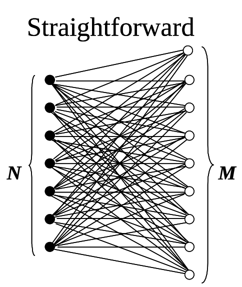
```
for m = 0,...,p-1
    c_m = 0;
    for i = 1,...,N
        c_m = c_m + a_m(x_i - x_*) u_i;
    end;
end;

for j = 1,...,M
    v_j = 0;
    for m = 0,...,p-1
        v_j = v_j + c_m f_m(y_j - x_*);
    end;
end;
```

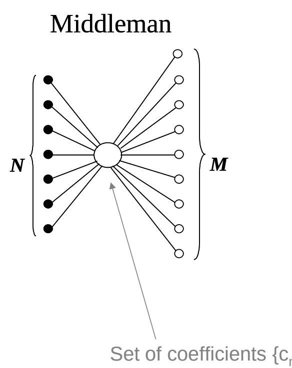
Complexity: $O(pN+pM)$

CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

Middleman Scheme



Complexity: $O(pN+pM)$



CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

Example Problem (1D Gauss Transform)

Compute

$$v_j = \sum_{i=1}^{N} \Phi(y_j, x_i) u_i, \quad j = 1, \ldots, M, \quad \Phi(y, x_i) = e^{-(y-x_i)^2}$$

where x_i, y_j, and u_i are random numbers distributed on [0,1].

Solution:

We have

$$\Phi(y, x_i) = e^{-(y-x_i)^2} = e^{-[y-x_*-(x_i-x_*)]^2} = e^{-(y-x_*)^2} e^{-(x_i-x_*)^2} e^{2(x_i-x_*)(y-x_*)}$$
$$= e^{-(y-x_*)^2} e^{-(x_i-x_*)^2} \left[\sum_{m=0}^{p-1} \frac{2^m (x_i - x_*)^m (y - x_*)^m}{m!} + error_p \right],$$
$$|error_p| \leqslant \frac{|y-x_*|^p}{p!} \sup_{0 \leqslant y \leqslant 1} \left| \frac{\partial^p e^{2(x_i-x_*)(y-x_*)}}{\partial y^p} \right| = \frac{2^p |y-x_*|^p |x_i - x_*|^p}{p!} \sup_{0 \leqslant y \leqslant 1} e^{2(x_i-x_*)(y-x_*)}.$$

Let us select $x_* = 0.5$, then truncation number $p = 10$ is sufficient for computations with $\epsilon = 10^{-6}$ and $N \leqslant 10^4$. The formula for fast computations will be then

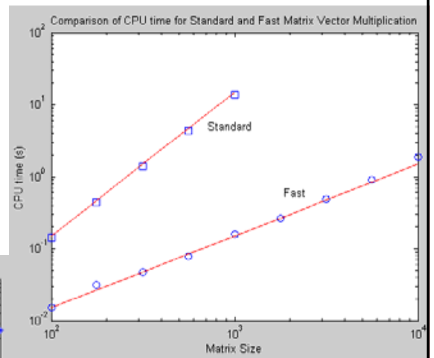
$$v_j = e^{-(y_j-x_*)^2} \sum_{m=0}^{p-1} c_m (y_j - x_*)^m, \quad j = 1, \ldots, M.$$
$$c_m = \frac{2^m}{m!} \sum_{i=1}^{N} e^{-(x_i-x_*)^2} (x_i - x_*)^m u_i.$$

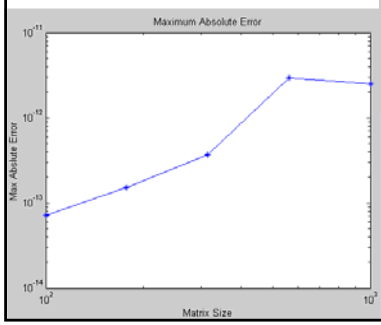
CS( )04

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Example Problem





Complexity of the Middleman Method

$$|error_p| \leqslant \sigma^{-p},$$
$$FMMerror_p \leqslant \sigma^{-p} N,$$
$$p \sim \log \frac{N}{\epsilon},$$
$$ComplexityFMM = O(pN) = O(N \log \frac{N}{\epsilon})$$

Local (Regular) Expansion

Let $\mathbf{x}_* \in \mathbb{R}^d$.

We call expansion

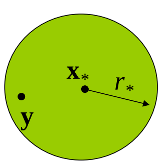
$$\Phi(\mathbf{y}, \mathbf{x}_i) = \sum_{m=0}^{\infty} a_m(\mathbf{x}_i, \mathbf{x}_*) R_m(\mathbf{y} - \mathbf{x}_*)$$

local (regular) inside a sphere $|\mathbf{y} - \mathbf{x}_*| < r_*$,

if the series converges for $\forall \mathbf{y}$, $|\mathbf{y} - \mathbf{x}_*| < r_*$.

Basis Functions

Expansion Coefficients



We also call this R-expansion, since basis functions R_m should be *regular*

Local Expansion (Example)

Valid for any $|x_i - x_*| > |y - x_*|$

$$x, y \in \mathbb{R}^1.$$

$$\Phi(y, x_i) = \frac{1}{y - x_i}.$$

Looking for factorization:

$$\Phi(y, x_i) = \sum_{m=0}^{\infty} a_m(x_i - x_*) R_m(y - x_*).$$

We have

$$\frac{1}{y - x_i} = \frac{1}{y - x_* - (x_i - x_*)} = -\frac{1}{(x_i - x_*)[1 - \frac{y - x_*}{x_i - x_*}]} = -\frac{1}{(x_i - x_*)}\left[1 - \frac{y - x_*}{x_i - x_*}\right]^{-1}.$$

Geometric progression:

$$(1 - \alpha)^{-1} = 1 + \alpha + \alpha^2 + \ldots = \sum_{m=0}^{\infty} \alpha^m, \quad |\alpha| < 1.$$

$$\left[1 - \frac{y - x_*}{x_i - x_*}\right]^{-1} = \sum_{m=0}^{\infty} \frac{(y - x_*)^m}{(x_i - x_*)^m}, \quad |y - x_*| < |x_i - x_*|.$$

Choose

$$a_m(x_i - x_*) = -\frac{1}{(x_i - x_*)^{m+1}}, \quad m = 0, 1, \ldots,$$
$$R_m(y - x_*) = (y - x_*)^m, \quad m = 0, 1, \ldots$$

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Example:

Let $\mathbf{x}_* \in \mathbb{R}^d$.

We call expansion

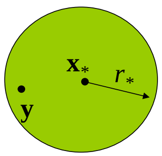
$$\Phi(\mathbf{y},\mathbf{x}_i) = \sum_{m=0}^{\infty} a_m(\mathbf{x}_i,\mathbf{x}_*) R_m(\mathbf{y}-\mathbf{x}_*)$$

local (regular) inside a sphere $|\mathbf{y}-\mathbf{x}_*| < r_*$,

if the series converges for $\forall \mathbf{y},\ |\mathbf{y}-\mathbf{x}_*| < r_*$.

Basis Functions

Expansion Coefficients



We also call this R-expansion, since basis functions R_m should be *regular*

Far Field (Singular) Expansions

Let $\mathbf{x}_* \in \mathrm{R}^d$.

Might be Singular (at $\mathbf{y} = \mathbf{x}_*$) Basis Functions

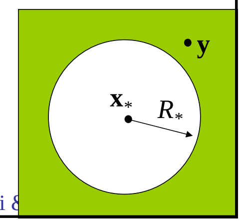
We call expansion

$$\Phi(\mathbf{y},\mathbf{x}_i) = \sum_{m=0}^{\infty} b_m(\mathbf{x}_i,\mathbf{x}_*) S_m(\mathbf{y}-\mathbf{x}_*)$$

far field expansion (or S-expansion) outside a sphere

$$|\mathbf{y}-\mathbf{x}_*| > R_*,$$

if the series converges for $\forall \mathbf{y},\ |\mathbf{y}-\mathbf{x}_*| > R_*$.



Example:

$$\Phi(y,x_i) = \frac{1}{y-x_i}.$$

$$\frac{1}{y-x_i} = \frac{1}{y-x_*-(x_i-x_*)} = \frac{1}{(y-x_*)\left[1-\frac{x_i-x_*}{y-x_*}\right]} = \frac{1}{(y-x_*)}\left[1-\frac{x_i-x_*}{y-x_*}\right]^{-1}.$$

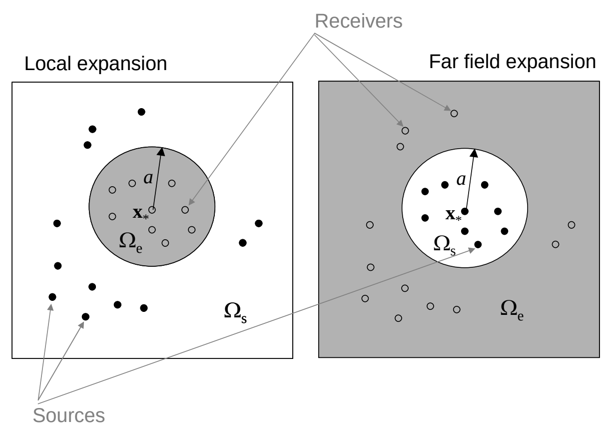
$$\left[1-\frac{x_i-x_*}{y-x_*}\right]^{-1} = \sum_{m=0}^{\infty}\frac{(x_i-x_*)^m}{(y-x_*)^m}, \quad |y-x_*| > |x_i-x_*|.$$

$$\Phi(y,x_i) = \sum_{m=0}^{\infty} b_m(x_i,x_*) S_m(y-x_*),$$

$$b_m(x_i,x_*) = (x_i-x_*)^m, \quad m = 0,1,\ldots,$$

$$S_m(y-x_*) = (y-x_*)^{-m-1}, \quad m = 0,1,\ldots$$

Middleman for Well Separated Domains:

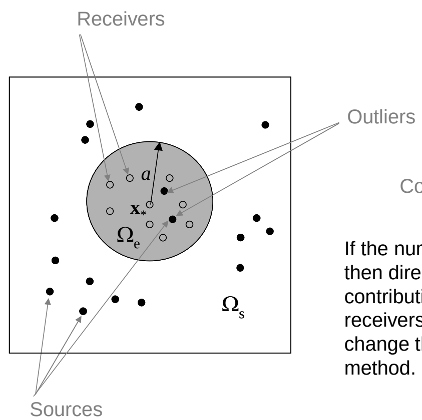


Complexity: $O(pN+pM)$

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

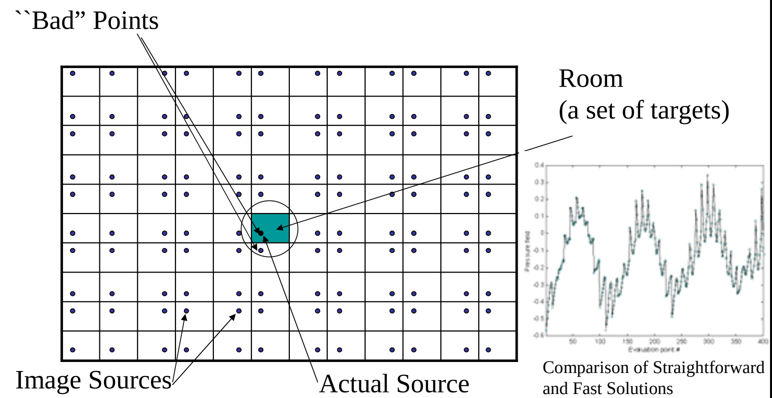
Problem with “Outliers”, or “Bad” Points



Complexity: $O(pN+pM)$

If the number of outliers is $O(p)$, then direct computation of their contribution to the field at M receivers is $O(pM)$, which does not change the complexity of the method.

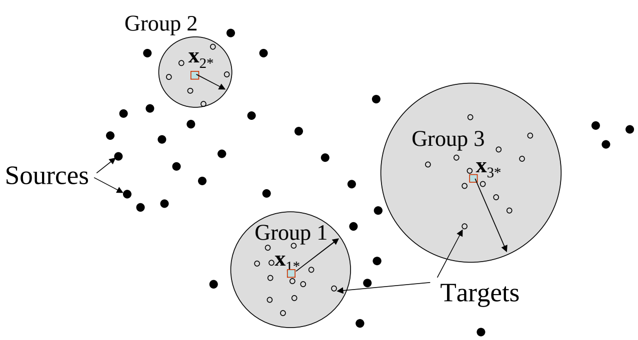
Example from Room Acoustics



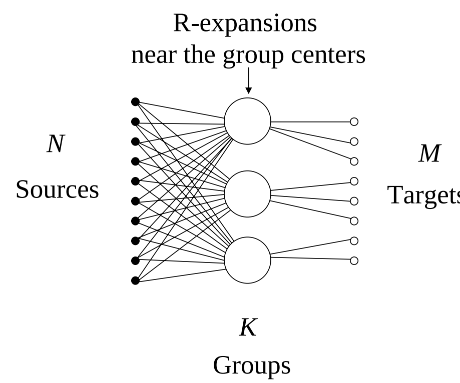
Comparison of Straightforward and Fast Solutions

(R. Duraiswami, N.A. Gumerov, D.N. Zotkin & L.S. Davis, Efficient Evaluation Of Reverberant Sound Fields, 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2001).

Natural Spatial Grouping for Well Separated Sets (Grouping with Respect to the Target Set)



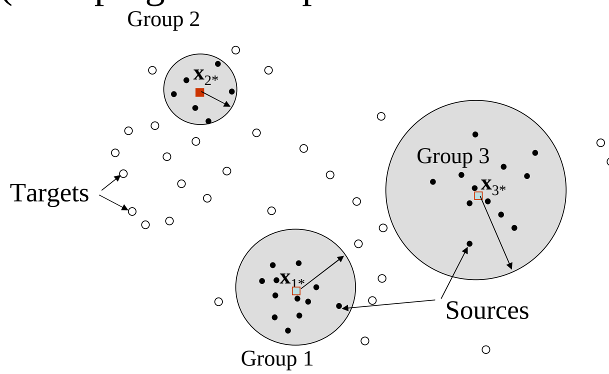
Natural Spatial Grouping for Well Separated Sets (continuation)



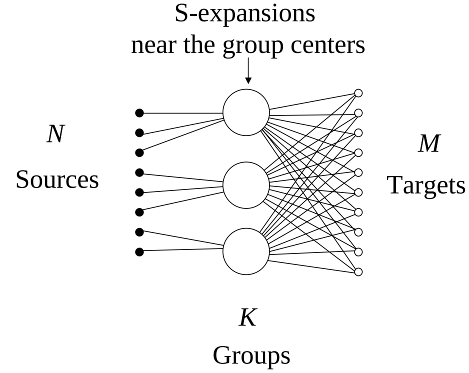
Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Natural Spatial Grouping for Well Separated Sets (Grouping with respect to the Source Set)



Natural Spatial Grouping for Well Separated Sets (continuation)



Examples of Natural Spatial Grouping

- Stars (Form Galaxies, Gravity);
- Flow Past a Body (Vortices are Grouped in a Wake);
- Statictics (Clusters of Statictical Data Points);
- People (Organized in Groups, Cities, etc.);
- Create your own example !

Space Partitioning
“Modified Middleman”

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

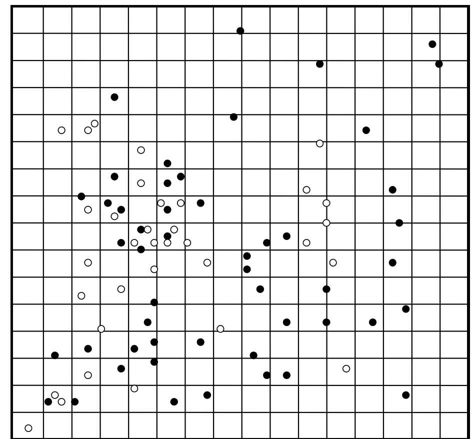
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Deficiencies of "Natural Grouping"

- Data points may be not naturally grouped;
- Need intelligence to identify the groups: Problem with the algorithms (Artificial Intelligence?)
- Problem dependent.

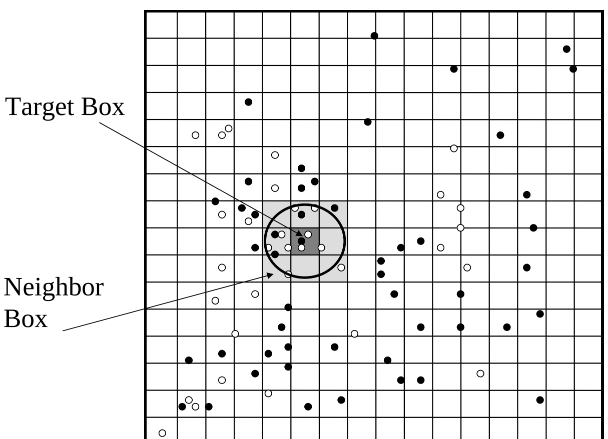
CSCAMM FAM04: 04/19/2004

The Answer Is: Space Partitioning



CSCAMM FAM04: 04/19/2004

Space Partitioning with Respect to the Target Set



CSCAMM FAM04: 04/19/2004

A Modified Middleman Algorithm

- Decomposition of the sum:

Singular Part (sources in the neighborhood)

$$v(\mathbf{y}_j) = \sum_{\mathbf{x}_i \in R_n^+} u_i \Phi(\mathbf{y}_j - \mathbf{x}_i) + \sum_{\mathbf{x}_i \in R_n^-} u_i \Phi(\mathbf{y}_j - \mathbf{x}_i), \quad \mathbf{y}_j \in R_n.$$

Regular Part (sources outside the neighborhood)

- Factorization of the regular part

$$\Phi(\mathbf{y}_j - \mathbf{x}_i) = \sum_{m=0}^{p-1} a_m(\mathbf{x}_i, \mathbf{x}_{n*}) R_m(\mathbf{y}_j - \mathbf{x}_{n*}) + Error_p, \quad \mathbf{y}_j, \mathbf{x}_{n*} \in R_n, \quad \mathbf{x}_i \in R_n^-.$$

- Fast computation of the regular part

$$\sum_{\mathbf{x}_i \in R_n^-} u_i \Phi(\mathbf{y}_j - \mathbf{x}_i) = \sum_{m=0}^{p-1} \left[\sum_{\mathbf{x}_i \in R_n^-} u_i a_m(\mathbf{x}_i, \mathbf{x}_{n*}) \right] R_m(\mathbf{y}_j - \mathbf{x}_{n*}).$$

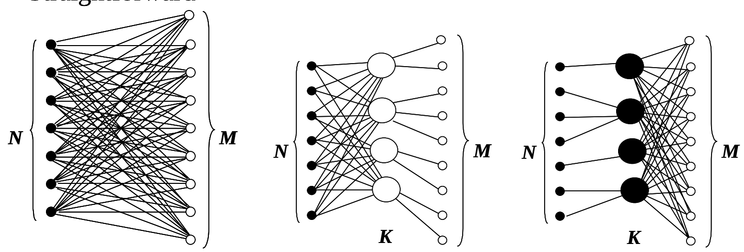
- Direct summation of the singular part, $\sum_{\mathbf{x}_i \in R_n^+} u_i \Phi(\mathbf{y}_j - \mathbf{x}_i)$

CSCAMM FAM04: 04/19/2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

A Scheme of “Modified Middleman”

Straightforward   Modified Middleman



A scheme with local expansions

A scheme with far field expansions

Asymptotic Complexity of the “Modified Middleman Method”

- Let N be the number of sources, M the number of targets, and K the number of target boxes.
- Each target box, R_n, contains M_n targets, $n = 1,\dots,K$.
- The *neighborhood* of each target box contains N_n sources, $n = 1,\dots,K$.
- Computation of the expansion coefficients for the regular part for the nth box requires $O((N-N_n)p)$ operations.
- Evaluation of the regular expansion for the nth box requires $O(M_n p)$ operations.
- Direct computation of the singular part requires $O(M_n N_n)$ operations.
- Total complexity is:

$$Complexity = O\left(\sum_{n=1}^{K}[(N-N_n)p + M_n p + M_n N_n]\right).$$

Asymptotic Complexity of the Modified Middleman (continued)

We have

$$\sum_{n=1}^{K} M_n = M$$

Power of the neighborhood of dimensionality d (the number of boxes in the neighborhood)

Consider a uniform distribution, then

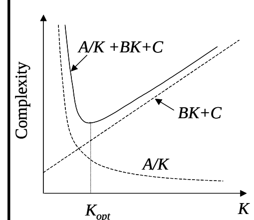
$$N_n \sim const \sim \frac{NPow(d)}{K},$$

$$F(K) = \sum_{n=1}^{K}[(N-N_n)p + M_n p + M_n N_n] = KNp - NpPow(d) + Mp + \frac{MNPow(d)}{K}$$

$$= \frac{MN}{K}Pow(d) + (K - Pow(d))Np + Mp$$

$$Complexity = O(F(K))$$

Optimization of the box number



Optimum complexity

$$F(K) = \frac{MN}{K}Pow(d) + (K - Pow(d))Np + Mp$$

$$K_{opt} = \left[\frac{MNPow(d)}{Np}\right]^{1/2} = \sqrt{\frac{MPow(d)}{p}}.$$

$$Complexity = O(F(K_{opt})) = O\left(Np\left(2\sqrt{\frac{MPow(d)}{p}} - Pow(d)\right) + Mp\right)$$

For $M \sim N,\ p \ll N$:

$$Complexity = O(N^{3/2}p^{1/2})$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Translations
Single Level FMM

Translations (Reexpansions)

Let $\{F_m(\mathbf{y}-\mathbf{x}_{*1})\}$ and $\{G_m(\mathbf{y}-\mathbf{x}_{*2})\}$ be two sets of basis functions centered at $\mathbf{x}_{*1}$ and $\mathbf{x}_{*2}$, such that $\Phi(\mathbf{y}_j,\mathbf{x}_i)$ can be represented by two absolutely and uniformly convergent series in domains Ω_1 and $\Omega_2 \subset \Omega_1$:

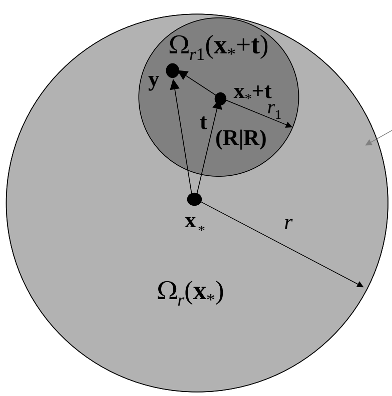
$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{n=0}^{\infty} a_n(\mathbf{x}_i-\mathbf{x}_{*1})F_n(\mathbf{y}_j-\mathbf{x}_{*1}), \quad \mathbf{y}_j \in \Omega_1$$

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{\infty} b_m(\mathbf{x}_i-\mathbf{x}_{*2})G_m(\mathbf{y}_j-\mathbf{x}_{*2}), \quad \mathbf{y}_j \in \Omega_2 \subset \Omega_1.$$

Under "translation" or "reexpansion" we mean an operator which relates the two sets of expansion coefficients:

$$\{b_m(\mathbf{x}_i-\mathbf{x}_{*2})\} = (F|G)(\mathbf{t})\{a_n(\mathbf{x}_i-\mathbf{x}_{*1})\}, \quad \mathbf{t} = \mathbf{x}_{*2}-\mathbf{x}_{*1}.$$

R|R-reexpansion (Local to Local, or L2L)

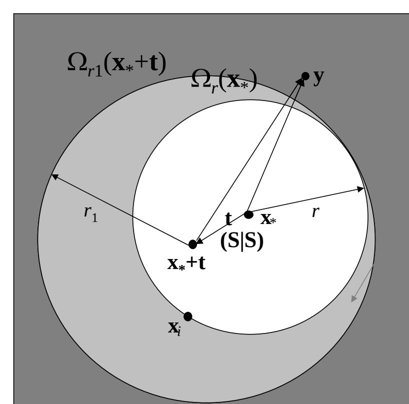


Original expansion
Is valid only here!

$|\mathbf{y} - \mathbf{x}_* - \mathbf{t}| < r_1 = r - |\mathbf{t}|$

Since $\Omega_{r1}(\mathbf{x}_*+\mathbf{t}) \subset \Omega_r(\mathbf{t})$!

S|S-reexpansion (Far to Far, or Multipole to Multipole, or M2M)



Original expansion
Is valid only here!

$|\mathbf{y} - \mathbf{x}_* - \mathbf{t}| > r_1 = r + |\mathbf{t}|$

Since
$\Omega_{r1}(\mathbf{x}_*+\mathbf{t}) \subset \Omega_r(\mathbf{t})$!

Also

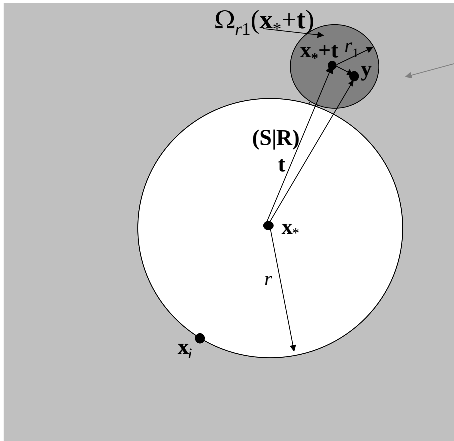
$|\mathbf{x}_i - \mathbf{x}_*| < r$

singular point !

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

S|R-reexpansion (Far to Local, or Multipole to Local, or M2L)



Original expansion Is valid only here!

$|\mathbf{y} - \mathbf{x}_* - \mathbf{t}| < r_1 = |\mathbf{t}| - r$

Since $\Omega_{r1}(\mathbf{x}_*+\mathbf{t}) \subset \Omega_r(\mathbf{t})$!

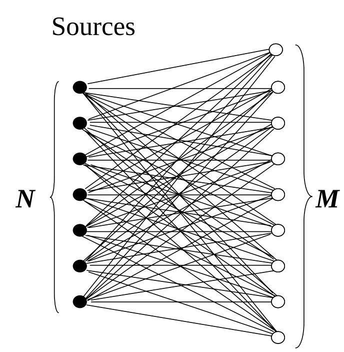
Also

$|\mathbf{x}_i - \mathbf{x}_*| < r$

singular point !

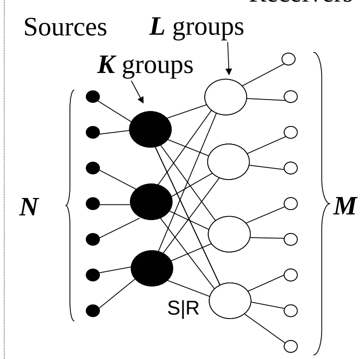
Single Level FMM

Standard algorithm



Total number of operations: ***O(NM)***

SLFMM

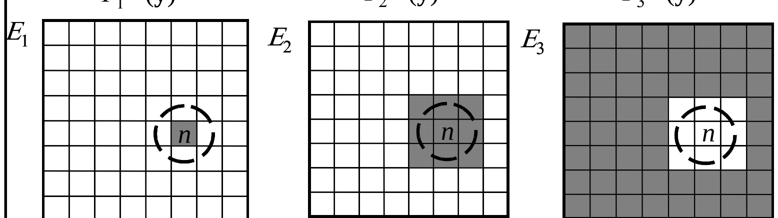


Total number of operations: ***O(N+M+KL)***

Spatial Domains

Potentials due to sources in these spatial domains

$\Phi_1^{(n)}(\mathbf{y})$ $\Phi_2^{(n)}(\mathbf{y})$ $\Phi_3^{(n)}(\mathbf{y})$



$I_1(n) = n$

$I_2(n) = \{Neighbors(n)\} \cup n$

$I_3(n) = \{All\ boxes\} \backslash I_2(n)$

Boxes with these numbers belong to these spatial domains

Definition of Potentials

$$\Phi_1^{(n)}(\mathbf{y}) = \sum_{\mathbf{x}_i \in E_1(n)} u_i \Phi(\mathbf{y}, \mathbf{x}_i),$$

$$\Phi_2^{(n)}(\mathbf{y}) = \sum_{\mathbf{x}_i \in E_2(n)} u_i \Phi(\mathbf{y}, \mathbf{x}_i),$$

$$\Phi_3^{(n)}(\mathbf{y}) = \sum_{\mathbf{x}_i \in E_3(n)} u_i \Phi(\mathbf{y}, \mathbf{x}_i),$$

Since domains $E_2(n)$ and $E_3(n)$ are complimentary:

$$\Phi(\mathbf{y}) = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}, \mathbf{x}_i) = \sum_{\mathbf{x}_i \in E_2(n) \cup E_3(n)} u_i \Phi(\mathbf{y}, \mathbf{x}_i) = \Phi_2^{(n)}(\mathbf{y}) + \Phi_3^{(n)}(\mathbf{y})$$

for arbitrary n.

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Step 1. Generate S-expansion coefficients for each box

$$\Phi_1^{(n)}(\mathbf{x}) = \mathbf{C}^{(n)} \circ \mathbf{S}(\mathbf{x} - \mathbf{x}_c^{(n)}),$$
$$\mathbf{C}^{(n)} = \sum_{\mathbf{x}_i \in E_1(n,L)} u_i \mathbf{B}\left(\mathbf{x}_i, \mathbf{x}_c^{(n)}\right).$$

For $n \in NonEmptySource$ ← loop over all non-empty source boxes

Get $\mathbf{x}_c^{(n)}$, the center of the box;

$\mathbf{C}^{(n)} = \mathbf{0}$;

For $\mathbf{x}_i \in E_1(n)$ ← loop over all sources in the box

Get $\mathbf{B}(\mathbf{x}_i, \mathbf{x}_c^{(n)})$, the S-expansion coefficients near the center of the box;

$\mathbf{C}^{(n)} = \mathbf{C}^{(n)} + u_i \mathbf{B}(\mathbf{x}_i, \mathbf{x}_c^{(n)})$;

End;

End;

Implementation can be different!
All we need is to get $\mathbf{C}^{(n)}$

Step 2. (S|R)-translate expansion coefficients

$$\Phi_3^{(n)}(\mathbf{y}) = \mathbf{D}^{(n)} \circ \mathbf{R}(\mathbf{y} - \mathbf{x}_c^{(n)}),$$
$$\mathbf{D}^{(n)} = \sum_{m \in I_3(n)} (\mathbf{S}|\mathbf{R})\left(\mathbf{x}_c^{(n)} - \mathbf{x}_c^{(m)}\right)\mathbf{C}^{(m)}.$$

For $n \in NonEmptyEvaluation$ ← loop over all non-empty evaluation boxes

Get $\mathbf{x}_c^{(n)}$, the center of the box;

$\mathbf{D}^{(n)} = \mathbf{0}$;

For $m \in I_3(n)$ ← loop over all non-empty source boxes outside the neighborhood of the *n*-th box

Get $\mathbf{x}_c^{(m)}$, the center of the box;

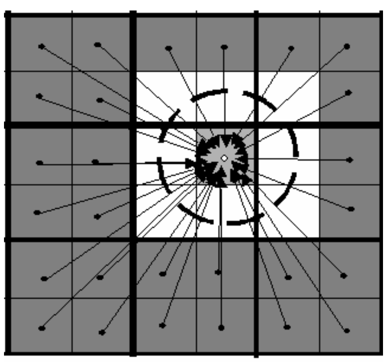
$\mathbf{D}^{(n)} = \mathbf{D}^{(n)} + (\mathbf{S}|\mathbf{R})(\mathbf{x}_c^{(n)} - \mathbf{x}_c^{(m)})\ \mathbf{C}^{(m)}$;

End;

End;

Implementation can be different!
All we need is to get $\mathbf{D}^{(n)}$

S|R-translation



Step 3. Final Summation

$$v_j = \Phi(\mathbf{y}_j) = \sum_{\mathbf{x}_i \in E_2(n)} \Phi(\mathbf{y}_j, \mathbf{x}_i) + \mathbf{D}^{(n)} \circ \mathbf{R}(\mathbf{y}_j - \mathbf{x}_c^{(n)}), \quad \mathbf{y}_j \in E_1(n).$$

For $n \in NonEmptyEvaluation$ ← loop over all boxes containing evaluation points

Get $\mathbf{x}_c^{(n)}$, the center of the box;

For $\mathbf{y}_j \in E_1(n)$ ← loop over all evaluation points in the box

$v_j = \mathbf{D}^{(n)} \circ \mathbf{R}(\mathbf{y}_j - \mathbf{x}_c^{(n)})$;

For $\mathbf{x}_i \in E_2(n)$ ← loop over all sources in the neighborhood of the *n*-th box

$v_j = v_j + \Phi(\mathbf{y}_j , \mathbf{x}_i)$;

End;

End;

End;

Implementation can be different!
All we need is to get v_j

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Asymptotic Complexity of SLFMM

Assume that:

- By some magic we can easily find neighbors, and lists of points in each box.
- Translation is performed by straightforward *P*x*P* matrix-vector multiplication, where $P(p)$ is the total length of the translation vector. So the complexity of a single translation is $O(P^2)$.
- The source and evaluation points are distributed uniformly, and there are K boxes, with s source points in each box ($s=N/K$). We call s the *grouping* (or *clustering*) parameter.
- The number of neighbors for each box is $O(1)$.

Then Complexity is:

- For Step 1: $O(PN)$
- For Step 2: $O(P^2K^2)$
- For Step 3: $O(PM+Ms)$
- Total: $O(PN+P^2K^2+PM+Ms) =$
  $O(PN+P^2K^2+PM+MN/K)$

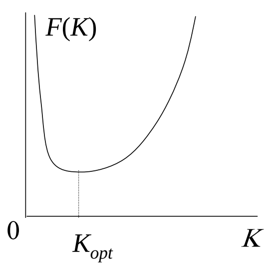
Selection of Optimal *K* (or *s*)

$$F(K) = PN + P^2K^2 + PM + PMN/K.$$

$$F'(K) = 2P^2K - PMN/K^2 = 0.$$

$$K_{opt} = \left(\frac{MN}{2P}\right)^{1/3} = O\left(\left(\frac{MN}{P}\right)^{1/3}\right).$$

$$s_{opt} = \frac{N}{K_{opt}} = \left(\frac{2PN^2}{M}\right)^{1/3} = O\left(\frac{PN^2}{M}\right)^{1/3}.$$



Complexity of Optimized SLFMM

$$F(K_{opt}) = PN + P^2\left(\frac{MN}{2P}\right)^{2/3} + PM + PMN\left(\frac{MN}{2P}\right)^{-1/3}$$
$$= P(M+N) + (MN)^{2/3}O(P^{4/3}).$$

At $K = K_{opt}$, and $M = O(N)$, the complexity of SLFMM is:

$$O(PN + P^{4/3}N^{4/3}) = O(P^{4/3}N^{4/3}).$$

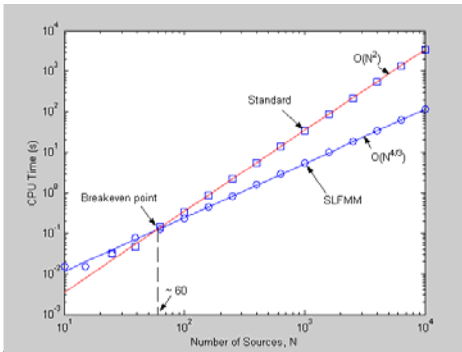
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Example of SLFMM

Compute matrix-vector product

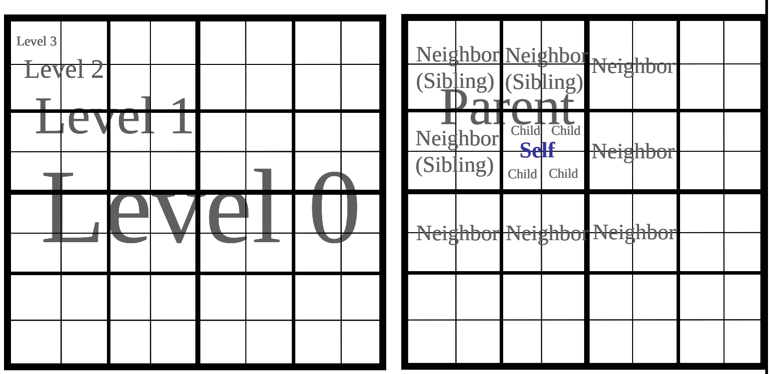
$$v_j = \sum_{i=1}^{N} \Phi_{ji} u_i, \quad j = 1,\dots,M, \quad \Phi_{ji} = \frac{1}{y_j - x_i},$$

where and $x_1,\dots,x_N$ are random points uniformly distributed on [0,10], $M = N-1$, and each y_j is located between the closest x_i's on each side, $j = 1,\dots,N-1$.

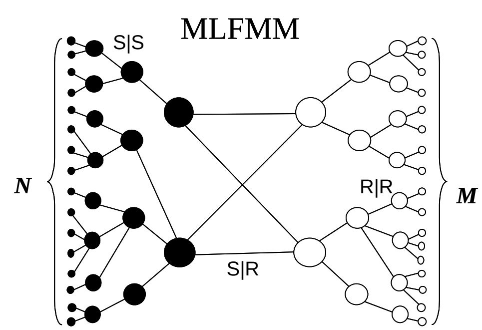


Hierarchical Space Partitioning (Multilevel FMM)

Hierarchy in 2^d-tree



A Scheme of MLFMM



Complexity = $O(pM+pN)$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Example of Multi Level Structure (Post Offices)

Source Hierarchy (Area)

- People (sources, Level 5)
- Mail Box, Post Master (Level 4)
- Local Post Offices (Level 3)
- City Post Office (Level 2)

Mail Transfer

AIRCRAFT

Receiver Hierarchy (Area)

- City Post Office (Level 2)
- Local Post Offices (Level 3)
- Post Master (Level 4)
- People (receivers, Level 5)

Mail Transfer

The MLFMM will be considered in more details in separate lectures

Representations and Translations of Functions in the FMM

Nail Gumerov &
Ramani Duraiswami
UMIACS
[gumerov][ramani]@umiacs.umd.edu

Content

- Function Representations and FMM Operations
- Matrix Representations of Translation Operators
- Integral Representations and Diagonal Forms of Translation Operators

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Function Representations and FMM Operations

What do we need in the FMM?

- Sum up functions;
- Translate functions (or represent them in different bases);
- In computations we can operate only with finite vectors.

Finite Approximations

Let

$$f : \mathbb{R}^d \to \mathbb{C} \quad (f = f(\mathbf{y}), \quad \mathbf{y} \in \mathbb{R}^d).$$

We consider *approximations* of $f(\mathbf{y})$ inside or outside a sphere $\Omega_a(\mathbf{x}_*)$ of radius a centered at $\mathbf{y} = \mathbf{x}_*$. We say that function $L_P(\mathbf{y};\mathbf{x}_*)$ *uniformly* approximate $f(\mathbf{y})$ inside a sphere $\Omega_a(\mathbf{x}_*)$ if

$$\exists \epsilon_P > 0, \quad \forall \mathbf{y} \in \Omega_a(\mathbf{x}_*) \subset \mathbb{R}^d, \quad |f(\mathbf{y}) - L_P(\mathbf{y};\mathbf{x}_*)| < \epsilon_P,$$

and function $F_P(\mathbf{y})$ uniformly approximate $f(\mathbf{y})$ outside a sphere $\Omega_a(\mathbf{x}_*)$ if

$$\exists \epsilon_P > 0, \quad \forall \mathbf{y} \notin \Omega_a(\mathbf{x}_*), \quad |f(\mathbf{y}) - F_P(\mathbf{y};\mathbf{x}_*)| < \epsilon_P.$$

The subscript P near functions $L_P(\mathbf{y};\mathbf{x}_*)$ and $F_P(\mathbf{y};\mathbf{x}_*)$ means that these functions can be determined by specification of a vector $\mathbf{C}$ in the complex P dimensional space $\mathbb{C}^P$, which we call *representing vector*.

So we have a one-to-one mapping of the space of functions $L_P(\mathbf{y};\mathbf{x}_*)$ to $\mathbf{C}(\mathbf{x}_*)$ and the space of functions $F_P(\mathbf{y};\mathbf{x}_*)$ to $\mathbf{C}(\mathbf{x}_*)$:

$$L_P(\mathbf{y},\mathbf{x}_*) \rightleftarrows \mathbf{C}(\mathbf{x}_*) = (c_1,\ldots,c_P), \quad \mathbf{C} \in \mathbb{C}^P,$$
$$F_P(\mathbf{y},\mathbf{x}_*) \rightleftarrows \mathbf{C}(\mathbf{x}_*) = (c_1,\ldots,c_P), \quad \mathbf{C} \in \mathbb{C}^P.$$

The representing vector $\mathbf{C}(\mathbf{x}_*)$ for $L_P(\mathbf{y};\mathbf{x}_*)$ we will identify as *local representation*. In the case when $\mathbf{C}(\mathbf{x}_*)$ corresponds to $F_P(\mathbf{y};\mathbf{x}_*)$ we call it as *far-field* representation.

Examples:

$P=p$ (real and complex functions)

Taylor expansion (for differentiable functions):

$$L_p(y;x_*) = \sum_{n=0}^{p-1} c_n (y - x_*)^n,$$
$$c_n = \frac{1}{n!} \frac{d^n f}{dy^n}\bigg|_{y=x_*}, \quad n = 0,\ldots,p-1.$$

Asymptotic expansion (for some decaying functions):

$$F_p(y;x_*) = \sum_{n=0}^{p-1} c_n (y-x_*)^{-n-1},$$
$$c_n = \lim_{y\to\infty} \left\{ (y-x_*)^{n+1} \left[f(y) - \sum_{m=0}^{n-1} c_m (y-x_*)^{-m-1} \right] \right\}, \quad n = 0,\ldots,p-1.$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Examples:

$P=p^2$ (Solutions of the Laplace equation in 3D)

$$L_p(\mathbf{y};\mathbf{x}_*) = \sum_{n=0}^{p-1}\sum_{m=-n}^{n} c_n r^n Y_n^m(\theta,\varphi),$$

$$F_p(\mathbf{y};\mathbf{x}_*) = \sum_{n=0}^{p-1}\sum_{m=-n}^{n} c_n r^{-n-1} Y_n^m(\theta,\varphi),$$

$$\mathbf{y}-\mathbf{x}_* = r(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$$

Examples:

$P=4N$ (Sum of Green's functions for Laplace equation in 3D)

$$L_P(\mathbf{y};\mathbf{x}_*) = \sum_{i=1}^{N}\frac{Q_i}{4\pi|\mathbf{y}-\mathbf{x}_i|}, \quad |\mathbf{x}_i-\mathbf{x}_*| > a,$$

$$F_P(\mathbf{y};\mathbf{x}_*) = \sum_{i=1}^{N}\frac{Q_i}{4\pi|\mathbf{y}-\mathbf{x}_i|}, \quad |\mathbf{x}_i-\mathbf{x}_*| > a,$$

$$\mathbf{C} = (x_{11}, x_{12}, x_{13}, Q_1, \ldots, x_{N1}, x_{N2}, x_{N3}, Q_N), \quad P = 4N.$$

Examples:

$P=N$ (Regular solution of the Helmholtz equation in 3D)

$$L_P(\mathbf{y};\mathbf{x}_*) = \sum_{j=1}^{N} w_j \Psi(\mathbf{s}_j) e^{ik\mathbf{s}_j\cdot(\mathbf{y}-\mathbf{x}_*)}, \; P = N,$$

$$\mathbf{C} = (w_1\Psi(\mathbf{s}_1), \ldots, w_N\Psi(\mathbf{s}_N))$$

follows from integral representation

$$f(\mathbf{y}) = \int_{S_u} e^{ik\mathbf{s}\cdot(\mathbf{y}-\mathbf{x}_*)}\Psi(\mathbf{s})dS.$$

Consolidation Operation

Linear operators (easy summation)

$$R_{P1}(\mathbf{y};\mathbf{x}_*) + R_{P2}(\mathbf{y};\mathbf{x}_*) \rightleftarrows \mathbf{C}_1(\mathbf{x}_*) + \mathbf{C}_2(\mathbf{x}_*).$$

Non-linear mapping (difficult summation):

$$R_{P1}(\mathbf{y};\mathbf{x}_*) + R_{P2}(\mathbf{y};\mathbf{x}_*) \rightleftarrows \mathbf{C}(\mathbf{x}_*) = \mathbf{C}_1(\mathbf{x}_*)[+]\mathbf{C}_2(\mathbf{x}_*).$$

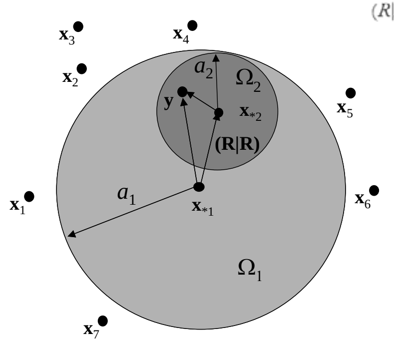
Consolidation operation

We usually focus on linear operators

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

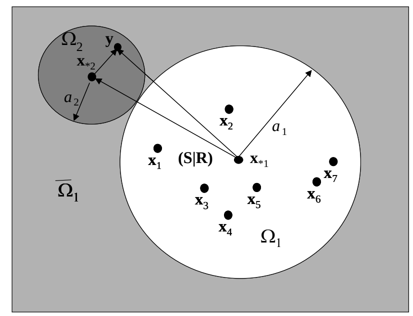
Translations: Local-to-local

$$(R|R)(\mathbf{x}_{*2} - \mathbf{x}_{*1})[\mathbf{C}_1(\mathbf{x}_{*1})] = \mathbf{C}_2(\mathbf{x}_{*2}).$$



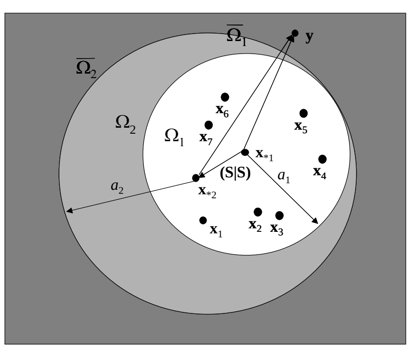
Translations: Multipole-to-local

$$(S|R)(\mathbf{x}_{*2} - \mathbf{x}_{*1})[\mathbf{C}_1(\mathbf{x}_{*1})] = \mathbf{C}_2(\mathbf{x}_{*2}).$$



Translations: Multipole-to-multipole

$$(S|S)(\mathbf{x}_{*2} - \mathbf{x}_{*1})[\mathbf{C}_1(\mathbf{x}_{*1})] = \mathbf{C}_2(\mathbf{x}_{*2}).$$



SLFMM in Terms of Representing Vectors:

- Subdivide the computational domain into N_b boxes.
- For each source $\mathbf{x}_i$ obtain vectors $\mathbf{C}_i\left(\mathbf{x}_{n*}^{(s)}\right)$ of length P_1 corresponding to function $S_{P_1}(\mathbf{y}, \mathbf{x}_{n*}^{(s)})$ approximating function $u_i\Phi(\mathbf{y}, \mathbf{x}_i)$ in the domain outer to the sphere $\Omega_a\left(\mathbf{x}_{n*}^{(s)}\right)$ (the sphere of radius a includes the box but enclosed into the box neighborhood) and $\mathbf{x}_{n*}^{(s)}$ is the center of the box containing $\mathbf{x}_i$.
- For each source box S_n containing q_n sources $\mathbf{x}_i, i = i_1, \dots, i_{q_n}$, obtain vector of length $P_2^{(n)}$ (consolidation of all sources inside the source box)

  $$\mathbf{C}\left(\mathbf{x}_{n*}^{(s)}\right) = \mathbf{C}_{i_1}\left(\mathbf{x}_{n*}^{(s)}\right)[+]\mathbf{C}_{i_2}\left(\mathbf{x}_{n*}^{(s)}\right)[+]\dots[+]\mathbf{C}_{i_{q_n}}\left(\mathbf{x}_{n*}^{(s)}\right).$$

  This vector represents potential due to all the sources inside the box in the domain outside the neighborhood of this box S_n^-.
- $S|R$ translate each $\mathbf{C}\left(\mathbf{x}_{n*}^{(s)}\right)$ from $\mathbf{x}_{n*}^{(s)}$ to the center $\mathbf{y}_{m*}^{(r)}$ of each receiver box R_m, such that the neighborhood of R_m does not contain S_n

  $$(S|R)\left(\mathbf{y}_{m*}^{(r)} - \mathbf{x}_{n*}^{(s)}\right)\left[\mathbf{C}\left(\mathbf{x}_{n*}^{(s)}\right)\right] = \mathbf{D}_n(\mathbf{y}_{m*}^{(r)}), \quad R_m^+ \cap S_n = \emptyset$$

  where $\mathbf{D}_n$ is the vector of length $P_3^{(n)}$ representing function in the domain inner to the sphere of radius a centered at $\mathbf{y}_{m*}^{(r)}$ due to sources in box S_n.
- For each receiver box R_m obtain vector (consolidation of all sources outside the receiver neighborhood)

  $$\mathbf{D}(\mathbf{y}_{m*}^{(r)}) = \mathbf{D}_{n_1}(\mathbf{y}_{m*}^{(r)})[+]\mathbf{D}_{n_2}(\mathbf{y}_{m*}^{(r)})[+]\dots[+]\mathbf{D}_{n_{p_m}}(\mathbf{y}_{m*}^{(r)}), \quad R_m^+ \cap S_{n_1}, \dots, S_{n_{p_m}} = \emptyset.$$
- For each receiver box evaluate the sum

  $$\psi(\mathbf{y}_j) = \sum_{\mathbf{x}_i \in R_m^+} u_i\Phi(\mathbf{y}_j, \mathbf{x}_i) + R_{P_4}(\mathbf{y}_j; \mathbf{y}_{m*}^{(r)}), \quad \mathbf{y}_j \in R_m.$$

  where $R_{P_4}(\mathbf{y}; \mathbf{y}_{m*}^{(r)})$ is the local function represented by $\mathbf{D}(\mathbf{y}_{m*}^{(r)})$ and R_m is the mth receiver box.

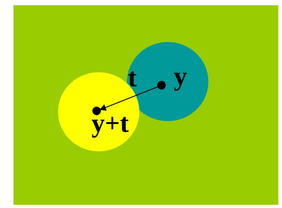
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Matrix Representations of Translation Operators

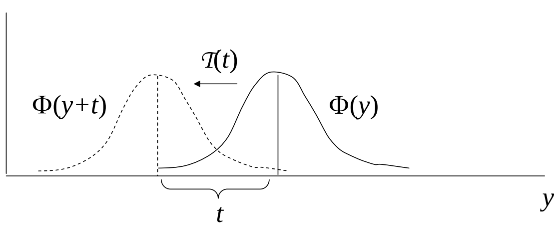
Translation Operator

Operator $\mathcal{T}(\mathbf{t}) : \mathbb{F}(\Omega) \to \mathbb{F}(\Omega'), \Omega' \subset \mathbb{R}^d, \quad \Omega \subset \mathbb{R}^d$ is called *translation* operator corresponding to *translation* vector $\mathbf{t}$, if

$$\mathcal{T}(\mathbf{t})[\Phi(\mathbf{y})] = \Phi(\mathbf{y}+\mathbf{t}), \quad (\mathbf{y} \in \Omega, \quad \mathbf{y}+\mathbf{t} \in \Omega').$$



Example of Translation Operator



R|R-reexpansion

Let $\mathbf{y}-\mathbf{x}_* \in \Omega_r(\mathbf{x}_*) \subset \mathbb{R}^d, \quad \Omega_r(\mathbf{x}_*) : |\mathbf{y}-\mathbf{x}_*| < r,$ and $\{R_n(\mathbf{y}-\mathbf{x}_*)\}$ be a regular basis in $C(\Omega)$. Let $\mathbf{y}-\mathbf{x}_*+\mathbf{t} \in \Omega_r(\mathbf{x}_*)$ and

$$R_n(\mathbf{y}-\mathbf{x}_*+\mathbf{t}) = \sum_{l=0}^{\infty} (R|R)_{ln}(\mathbf{t}) R_l(\mathbf{y}-\mathbf{x}_*).$$

Coefficients $(R|R)_{ln}(\mathbf{t})$ are called $R|R-$ *reexpansion coefficients* (regular-to-regular) , and infinite matrix

$$(\mathbf{R}|\mathbf{R})(\mathbf{t}) = \begin{pmatrix} (R|R)_{00} & (R|R)_{01} & \dots \\ (R|R)_{10} & (R|R)_{11} & \dots \\ \dots & \dots & \dots \end{pmatrix}$$

is called $R|R-$ *reexpansion matrix*.

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Example of R|R-reexpansion

$$R_m(x) = x^m,$$

$$R_m(x+t) = (x+t)^m = x^m + \binom{m}{1} x^{m-1} t + \ldots + \binom{m}{m-1} x t^{m-1} + t^m$$

$$= \sum_{l=0}^{m} \binom{m}{l} t^l x^{m-l} = \sum_{l=0}^{m} \binom{m}{l} t^{m-l} x^l = \sum_{l=0}^{m} \binom{m}{l} t^{m-l} R_l(x),$$

$$(R|R)_{lm}(t) = \begin{cases} \binom{m}{l} t^{m-l}, & l \leqslant m, \\ 0, & l > m. \end{cases}$$

R|R-translation operator

Translation operator $\mathcal{T}(\mathbf{t})$ which is represented in regular basis $\{R_n(\mathbf{y}-\mathbf{x}_*)\}$ by the $R|R$ – *reexpansion matrix* is called $\mathcal{R}|\mathcal{R}$-translation operator.

$$\mathcal{T}(\mathbf{t})[\Phi(\mathbf{y})] = \Phi(\mathbf{y}+\mathbf{t})$$

$$(\mathcal{R}|\mathcal{R})(\mathbf{t}) = \mathcal{T}(\mathbf{t}).$$

CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

Why the same operator named differently?

$$\mathcal{T}(\mathbf{t})[\Phi(\mathbf{y})] = \Phi(\mathbf{y}+\mathbf{t})$$

The first letter shows the basis for $\Phi(\mathbf{y})$

The second letter shows the basis for $\Phi(\mathbf{y}+\mathbf{t})$

$$\mathcal{T}(\mathbf{t}) = \begin{cases} (\mathcal{R}|\mathcal{R})(\mathbf{t}) \\ (\mathcal{S}|\mathcal{S})(\mathbf{t}) \\ (\mathcal{S}|\mathcal{R})(\mathbf{t}) \\ (\mathcal{R}|\mathcal{S})(\mathbf{t}) \end{cases}$$

Needed only to show the expansion basis (for operator representation)

CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

Matrix representation of R|R-translation operator

Consider $\Phi(\mathbf{y}) = \sum_{n=0}^{\infty} A_n(\mathbf{x}_*) R_n(\mathbf{y}-\mathbf{x}_*).$

$$\Phi(\mathbf{y}+\mathbf{t}) = (\mathcal{R}|\mathcal{R})(\mathbf{t})[\Phi(\mathbf{y})] = \sum_{n=0}^{\infty} A_n(\mathbf{x}_*)(\mathcal{R}|\mathcal{R})(\mathbf{t})[R_n(\mathbf{y}-\mathbf{x}_*)]$$

$$= \sum_{n=0}^{\infty} A_n(\mathbf{x}_*) R_n(\mathbf{y}-\mathbf{x}_*+\mathbf{t})$$

$$= \sum_{n=0}^{\infty} A_n(\mathbf{x}_*) \sum_{l=0}^{\infty} (R|R)_{ln}(\mathbf{t}) R_l(\mathbf{y}-\mathbf{x}_*)$$

$$= \sum_{l=0}^{\infty} \left[\sum_{n=0}^{\infty} (R|R)_{ln}(\mathbf{t}) A_n(\mathbf{x}_*) \right] R_l(\mathbf{y}-\mathbf{x}_*)$$

$$= \sum_{l=0}^{\infty} \widetilde{A}_l(\mathbf{x}_*, \mathbf{t}) R_l(\mathbf{y}-\mathbf{x}_*),$$

Coefficients of shifted function

Coefficients of original function

$$\widetilde{A}_l(\mathbf{x}_*, \mathbf{t}) = \sum_{n=0}^{\infty} (R|R)_{ln}(\mathbf{t}) A_n(\mathbf{x}_*), \quad \widetilde{\mathbf{A}}(\mathbf{x}_*, \mathbf{t}) = (\mathbf{R}|\mathbf{R})(\mathbf{t}) \mathbf{A}(\mathbf{x}_*).$$

CSCAMM FAM
2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Reexpansion of the same function over shifted basis

Compact notation:

$$\Phi(\mathbf{y}) = \sum_{n=0}^{\infty} A_n(\mathbf{x}_*) R_n(\mathbf{y}-\mathbf{x}_*) = \mathbf{A}(\mathbf{x}_*) \circ \mathbf{R}(\mathbf{y}-\mathbf{x}_*),$$

$$\Phi(\mathbf{y}+\mathbf{t}) = \sum_{l=0}^{\infty} \widetilde{A}_l(\mathbf{x}_*,\mathbf{t}) R_l(\mathbf{y}-\mathbf{x}_*) = \widetilde{\mathbf{A}}(\mathbf{x}_*,\mathbf{t}) \circ \mathbf{R}(\mathbf{y}-\mathbf{x}_*)$$

We have:

$$\Phi(\mathbf{y}) = \Phi((\mathbf{y}-\mathbf{t})+\mathbf{t}) = \widetilde{\mathbf{A}}(\mathbf{x}_*,\mathbf{t}) \circ \mathbf{R}((\mathbf{y}-\mathbf{t})-\mathbf{x}_*)$$
$$= \widetilde{\mathbf{A}}(\mathbf{x}_*,\mathbf{t}) \circ \mathbf{R}(\mathbf{y}-\mathbf{x}_*-\mathbf{t}).$$

Also

$$\Phi(\mathbf{y}) = \mathbf{A}(\mathbf{x}_*) \circ \mathbf{R}(\mathbf{y}-\mathbf{x}_*) = \mathbf{A}(\mathbf{x}_*+\mathbf{t}) \circ \mathbf{R}(\mathbf{y}-\mathbf{x}_*-\mathbf{t}),$$

so

$$\mathbf{A}(\mathbf{x}_*+\mathbf{t}) = \widetilde{\mathbf{A}}(\mathbf{x}_*,\mathbf{t}) = (\mathbf{R}|\mathbf{R})(\mathbf{t})\mathbf{A}(\mathbf{x}_*).$$

Example

$$\Phi(y,x_i) = \frac{1}{y-x_i}.$$

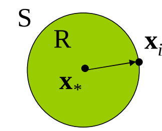
$|y-x_*| < |x_i-x_*|$:

R-expansion

$$\Phi(y,x_i) = \sum_{m=0}^{\infty} a_m(x_i,x_*) R_m(y-x_*),$$

$$a_m(x_i,x_*) = -(x_i-x_*)^{-m-1}, \quad m = 0,1,\ldots,$$

$$R_m(y-x_*) = (y-x_*)^m, \quad m = 0,1,\ldots$$



$|y-x_*| > |x_i-x_*|$:

S-expansion

$$\Phi(y,x_i) = \sum_{m=0}^{\infty} b_m(x_i,x_*) S_m(y-x_*),$$

$$b_m(x_i,x_*) = (x_i-x_*)^m, \quad m = 0,1,\ldots,$$

$$S_m(y-x_*) = (y-x_*)^{-m-1}, \quad m = 0,1,\ldots$$

R|R-operator

$$R_n(y+t) = (y+t)^n = \sum_{m=0}^{n} \frac{n!}{m!(n-m)!} t^{n-m} y^m = \sum_{m=0}^{n} \frac{n!}{m!(n-m)!} t^{n-m} R_m(y).$$

$$(R|R)_{mn}(t) = \begin{cases} 0, & m > n \\ \frac{n!}{m!(n-m)!} t^{n-m}, & m \leqslant n \end{cases}.$$

$$(\mathbf{R}|\mathbf{R})(t) = (R|R)_{mn}(t) = \begin{pmatrix} 1 & t & t^2 & t^3 & \ldots \\ 0 & 1 & 2t & 3t^2 & \ldots \\ 0 & 0 & 1 & 3t & \ldots \\ 0 & 0 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix}.$$

S|S-operator

$$S_n(y+t) = (y+t)^{-n-1} = y^{-n-1}\left(1+\frac{t}{y}\right)^{-n-1} = \sum_{m=n}^{\infty} \frac{(-1)^{m-n} m!}{n!(m-n)!} t^{m-n} S_m(y),$$

$$(S|S)_{mn}(t) = \begin{cases} 0, & m < n \\ \frac{(-1)^{m-n} m!}{n!(m-n)!} t^{m-n}, & m \geqslant n. \end{cases}.$$

$$(\mathbf{S}|\mathbf{S})(t) = (S|S)_{mn}(t) = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots \\ -t & 1 & 0 & 0 & \ldots \\ t^2 & -2t & 1 & 0 & \ldots \\ -t^3 & 3t^2 & -3t & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix}.$$

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

S|R-operator

$$S_n(y+t) = (t+y)^{-n-1} = t^{-n-1}\left(1+\frac{y}{t}\right)^{-n-1} = \sum_{m=0}^{\infty} \frac{(-1)^m (m+n)!}{m!n!} t^{-n-m-1} y^m$$

$$= \sum_{m=0}^{\infty} \frac{(-1)^m (m+n)!}{m!n!} t^{-n-m-1} R_m(y).$$

$$(S|R)_{mn}(t) = \frac{(-1)^m (m+n)!}{m!n!t^{n+m+1}},$$

$$(\mathbf{S}|\mathbf{R})(t) = \begin{pmatrix} t^{-1} & t^{-2} & t^{-3} & \ldots \\ -t^{-2} & -2t^{-3} & -3t^{-4} & \ldots \\ t^{-3} & 3t^{-4} & 6t^{-5} & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}.$$

Renormalized R-functions

$$\widetilde{R}_n(y) = \frac{y^n}{n!}.$$

Then

$$\widetilde{R}_n(y+t) = \frac{1}{n!}(y+t)^n = \frac{1}{n!}\sum_{m=0}^{n} \frac{n!}{m!(n-m)!} t^{n-m} y^m = \sum_{m=0}^{n} \widetilde{R}_{n-m}(t)\widetilde{R}_m(y).$$

$$\left(\widetilde{R}|\widetilde{R}\right)_{mn}(t) = \begin{cases} 0, & m > n \\ \frac{1}{(n-m)!} t^{n-m} = \widetilde{R}_{n-m}(t), & m \leqslant n \end{cases}.$$

Translation Matrix:

$$\left(\widetilde{\mathbf{R}}|\widetilde{\mathbf{R}}\right)(t) = \left(\widetilde{R}|\widetilde{R}\right)_{mn}(t) = \begin{pmatrix} 1 & t & \frac{t^2}{2} & \frac{t^3}{6} & \ldots \\ 0 & 1 & t & \frac{t^2}{2} & \ldots \\ 0 & 0 & 1 & t & \ldots \\ 0 & 0 & 0 & 1 & \ldots \end{pmatrix}.$$

Toeplitz

Renormalized S-functions

$$\widetilde{S}_n(y) = \frac{(-1)^n n!}{y^{n+1}}.$$

$$\widetilde{S}_n(y+t) = (-1)^n n!(y+t)^{-n-1} = (-1)^n n! \sum_{m=n}^{\infty} \frac{(-1)^{m-n} m!}{n!(m-n)!} t^{m-n} y^{-m-1} = \sum_{m=n}^{\infty} \widetilde{R}_{m-n}(t)\widetilde{S}_m(y).$$

$$\left(\widetilde{S}|\widetilde{S}\right)_{mn}(t) = \begin{cases} 0, & m < n \\ \frac{1}{(m-n)!} t^{m-n} = \widetilde{R}_{m-n}(t), & m \geqslant n \end{cases}.$$

Translation Matrix:

Toeplitz

$$\left(\widetilde{\mathbf{S}}|\widetilde{\mathbf{S}}\right)(t) = \left(\widetilde{S}|\widetilde{S}\right)_{mn}(t) = \begin{pmatrix} 1 & 0 & 0 & 0 & \ldots \\ t & 1 & 0 & 0 & \ldots \\ \frac{t^2}{2} & t & 1 & 0 & \ldots \\ \frac{t^3}{6} & \frac{t^2}{2} & t & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix} = \left(\widetilde{\mathbf{R}}|\widetilde{\mathbf{R}}\right)^T(t).$$

Renormalized S-functions

$$\widetilde{S}_n(y+t) = \sum_{m=n}^{\infty} \widetilde{R}_{m-n}(y)\widetilde{S}_m(t) = \sum_{m=0}^{\infty} \widetilde{S}_{m+n}(t)\widetilde{R}_m(y).$$

$$\left(\widetilde{S}|\widetilde{R}\right)_{mn}(t) = \widetilde{S}_{m+n}(t).$$

Hankel

Translation Matrix:

$$\left(\widetilde{\mathbf{S}}|\widetilde{\mathbf{R}}\right)(t) = \left(\widetilde{S}|\widetilde{R}\right)_{mn}(t) = \begin{pmatrix} t^{-1} & -t^{-2} & 2t^{-3} & -6t^{-4} & \ldots \\ -t^{-2} & 2t^{-3} & -6t^{-4} & 24t^{-5} & \ldots \\ 2t^{-3} & -6t^{-4} & 24t^{-5} & -120t^{-6} & \ldots \\ -6t^{-4} & 24t^{-5} & -120t^{-6} & 720t^{-7} & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix}.$$

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Integral Representations and Diagonal Forms of Translation Operators

With such renormalized functions all translations can be performed with complexity O($p\log p$).

But we look for something faster.

Theoretical limit for translation of vector of length p is $O(p)$.

ONLY SPARSE TRANSLATION MATRIX CAN PROVIDE SUCH COMPLEXITY

Representations Based on Signature Functions

Definition *Let*

$$\Phi(y) = \sum_{m=0}^{\infty} C_m \widetilde{R}_m(y),$$

then the Signature Function of $\Phi(y)$ *is a* 2π*-periodic funcion*

$$\Phi^*(s) = \sum_{m=0}^{\infty} C_m e^{ims}.$$

.

Definition *Let*

$$\Phi(y) = \sum_{m=0}^{\infty} C_m \widetilde{S}_m(y),$$

then the Signature Function of $\Phi(y)$ *is a* 2π*-periodic funcion*

$$\Phi^*(s) = \sum_{m=0}^{\infty} C_m e^{-ims}.$$

We assume that series for SF converge. This is always true for finite series, $C_m = 0$, $m > p$-1.

Integral Representation of Regular Functions

$$C_m = \frac{1}{2\pi}\int_0^{2\pi} \Phi^*(s) e^{-ims} ds.$$ ← Property of Fourier coefficients

We have then the following representation of $\Phi(y)$:

$$\Phi(y) = \sum_{m=0}^{\infty} \widetilde{R}_m(y) \frac{1}{2\pi}\int_0^{2\pi} \Phi^*(s) e^{-ims} ds = \frac{1}{2\pi}\int_0^{2\pi} \Phi^*(s) \sum_{m=0}^{\infty} \widetilde{R}_m(y) e^{-ims} ds$$

Consider

$$\sum_{m=0}^{\infty} \widetilde{R}_m(y) e^{-ims} = \sum_{m=0}^{\infty} e^{-ims}\frac{y^m}{m!} = \sum_{m=0}^{\infty} \frac{\left(ye^{-is}\right)^m}{m!} = e^{ye^{-is}}.$$

So

$$\Phi(y) = \frac{1}{2\pi}\int_0^{2\pi} e^{ye^{-is}} \Phi^*(s) ds = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_r(y;s) \Phi^*(s) ds,$$

where

$$\Lambda_r(y;s) = e^{ye^{-is}}.$$ ← Regular kernel

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Integral Representation of Regular Basis Functions

For $\Phi(y) = \widetilde{R}_m(y)$ we have

$$\Phi(y) = \widetilde{R}_m(y) = \sum_{m'=0}^{\infty} C_{m'} \widetilde{R}_{m'}(y), \qquad C_{m'} = \delta_{mm'}$$

Therefore the SF for this function is

$$\Phi^*(s) = \sum_{m'=0}^{\infty} C_{m'} e^{im's} = \sum_{m'=0}^{\infty} \delta_{mm'} e^{im's} = e^{ims}.$$

Then

$$\widetilde{R}_m(y) = \Phi(y) = \frac{1}{2\pi}\int_0^{2\pi} e^{ye^{-is}} \Phi^*(s)ds = \frac{1}{2\pi}\int_0^{2\pi} e^{ye^{-is}} e^{ims} ds = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_r(y,s) e^{ims} ds.$$

Integral Representation of Singular Functions

$$\Phi^{(p)}(y) = \sum_{m=0}^{p-1} C_m \widetilde{S}_m(y), \qquad \Phi^{(p)*}(s) = \sum_{m=0}^{p-1} C_m e^{-ims}.$$

$$C_m = \frac{1}{2\pi}\int_0^{2\pi} \Phi^{(p)*}(s) e^{ims} ds.$$ ← Property of Fourier coefficients

We have then the following representation of $\Phi(y)$:

$$\Phi^{(p)}(y) = \sum_{m=0}^{p-1} \widetilde{S}_m(y) \frac{1}{2\pi}\int_0^{2\pi} \Phi^{(p)*}(s) e^{ims} ds = \frac{1}{2\pi}\int_0^{2\pi} \Phi^{(p)*}(s) \sum_{m=0}^{p-1} \widetilde{S}_m(y) e^{ims} ds$$

Then

$$\boxed{\Phi^{(p)}(y) = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_s^{(p)}(y;s)\Phi^{(p)*}(s)ds,}$$

$$\Lambda_s^{(p)}(y;s) = \sum_{m=0}^{p-1} \widetilde{S}_m(y) e^{ims} = \sum_{m=0}^{p-1} e^{ims} \frac{(-1)^m m!}{y^{m+1}}.$$ ← Singular kernel

Integral Representation of Singular Basis Functions

For $\Phi(y) = \widetilde{S}_m(y)$ we have

$$\Phi^{(p)}(y) = \widetilde{S}_m(y) = \sum_{m'=0}^{p-1} C_{m'} \widetilde{S}_{m'}(y), \qquad C_{m'} = \delta_{mm'}, \qquad p > m.$$

Therefore the SF for this function is

$$\Phi^{(p)*}(s) = \sum_{m'=0}^{\infty} C_{m'} e^{-im's} = \sum_{m'=0}^{\infty} \delta_{mm'} e^{-im's} = e^{-ims}, \qquad p > m.$$

Then

$$\widetilde{S}_m(y) = \Phi^{(p)}(y) = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_s^{(p)}(y,s)\Phi^{(p)*}(s)ds = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_s^{(p)}(y,s) e^{-ims} ds,$$

$$m < p.$$

R|R-translation of the Signature Function

$$\mathcal{T}(t)[\Phi(y)] = \Phi(y+t) = \frac{1}{2\pi}\int_0^{2\pi} e^{(y+t)e^{-is}} \Phi^*(s)ds = \frac{1}{2\pi}\int_0^{2\pi} e^{ye^{-is}} e^{te^{-is}} \Phi^*(s)ds$$

$$= \frac{1}{2\pi}\int_0^{2\pi} \Lambda_r(y,s)\Lambda_r(t,s)\Phi^*(s)ds = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_r(y,s)\widehat{\Phi}^*(s,t)ds.$$

$$\boxed{(\mathcal{R}|\mathcal{R})(t)[\Phi^*(s)] = \widehat{\Phi}^*(s,t) = \Lambda_r(t,s)\Phi^*(s).}$$

So the R|R translation of the SF means simply multiplication of the SF by the regular kernel !

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

S|S-translation of the Signature Function

$$\Phi^{(p)}(y+t) = \sum_{m=0}^{p-1} \hat{C}_m \tilde{S}_m(y) = \sum_{m=0}^{p-1}\sum_{n=0}^{\infty} \left(\tilde{S}|\tilde{S}\right)_{mn}(t) C_n \tilde{S}_m(y)$$

$$= \sum_{m=0}^{p-1}\sum_{n=0}^{p-1} \tilde{R}_{m-n}(t) C_n \tilde{S}_m(y) = \sum_{m=0}^{p-1}\sum_{n=0}^{p-1} \underbrace{\frac{1}{2\pi}\int_0^{2\pi} e^{te^{-is}} e^{i(m-n)s} ds}\, C_n \tilde{S}_m(y)$$

Representation of the regular basis function

$$= \frac{1}{2\pi}\int_0^{2\pi} e^{te^{-is}} \sum_{n=0}^{p-1} C_n e^{-ins} \sum_{m=0}^{p-1} \tilde{S}_m(y) e^{ims} ds$$

$$= \frac{1}{2\pi}\int_0^{2\pi} \Lambda_s^{(p)}(y,s) e^{te^{-is}} \Phi^{(p)*}(s) ds, \qquad |t| < |y|.$$

So

$$(\mathcal{S}|\mathcal{S})(t)[\Phi^{(p)*}(s)] = \hat{\Phi}^{(p)*}(s;t) = e^{te^{-is}}\Phi^{(p)*}(s) = \Lambda_r(t,s)\Phi^{(p)*}(s).$$

So the S|S translation of the SF means multiplication of the SF by the regular kernel.

S|R-translation of the Signature Function

In case $|t| > |y|$ we have

$$\Phi^{(p)}(y+t) = \frac{1}{2\pi}\int_0^{2\pi} e^{ye^{-is}} \Lambda_s^{(p)}(t,s) \Phi^{(p)*}(s) ds, \qquad |t| > |y|.$$

This is a representation of the regular function. Therefore,

$$(\mathcal{S}|\mathcal{R})(t)[\Phi^{(p)*}(s)] = \hat{\Phi}^{(p)*}(s;t) = \Lambda_s^{(p)}(t,s)\Phi^{(p)*}(s).$$

So the S|S translation of the SF means multiplication of the SF by the singular kernel.

Evaluation of Function based on its Signature Function

Use Gaussian Type Quadrature

function bandwidth

$$\Phi^{(p)}(y) = \frac{1}{2\pi}\int_0^{2\pi} \Lambda_r(y,s)\Phi^{(p)*}(s) ds$$

$$= \sum_{k=0}^{q-1} w_k \Lambda_r(y,s_k)\Phi^{(p)*}(s_k) + error(p,q).$$

quadrature weights

quadrature nodes

quadrature order

FMM Data Structures

Nail Gumerov &
Ramani Duraiswami
UMIACS
[gumerov][ramani]@umiacs.umd.edu

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Content

- Introduction
- Hierarchical Space Subdivision with 2^d-Trees
- Hierarchical Indexing System
  - Parent & Children Finding
- Binary Ordering
- Spatial Ordering Using Bit Interleaving
  - Neighbor & Box Center Finding
- Spatial Data Structuring
  - Threshold Level of Space Subdivision
- Operations on Sets

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Reference:

N.A. Gumerov, R. Duraiswami & E.A. Borovikov

Data Structures, Optimal Choice of Parameters, and Complexity Results for Generalized Multilevel Fast Multipole Methods in *d* Dimensions

UMIACS TR 2003-28, Also issued as *Computer Science Technical Report CS-TR-# 4458*. Volume 91 pages.

University of Maryland, College Park, 2003.

AVAILABLE ONLINE VIA http://www.umiacs.umd.edu/~gumerov
http://www.umiacs.umd.edu/~ramani

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Introduction

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

FMM Data Structures

- Since the complexity of FMM should not exceed $O(N^2)$ (at $M \sim N$), data organization should be provided for efficient numbering, search, and operations with these data.
- Some naive approaches can utilize search algorithms that result in $O(N^2)$ complexity of the FMM (and so they kill the idea of the FMM).
- In *d*-dimensions $O(N \log N)$ complexity for operations with data can be achieved.

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

FMM Data Structures (2)

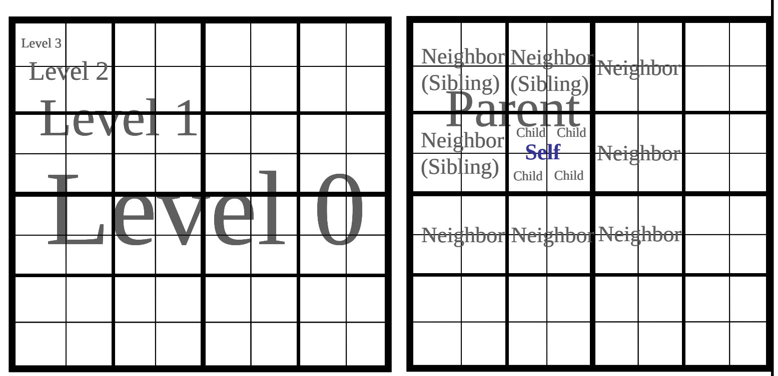
- Approaches include:
  - Data preprocessing
    - Sorting
    - Building lists (such as neighbor lists): requires memory, potentially can be avoided;
    - Building and storage of trees: requires memory, potentially can be avoided;
  - Operations with data during the FMM algorithm execution:
    - Operations on data sets;
    - Search procedures.
- Preferable algorithms:
  - Avoid unnecessary memory usage;
  - Use fast (constant and logarithmic) search procedures;
  - Employ bitwise operations;
  - Can be parallelized.
- Tradeoff Between Memory and Speed

Space Subdivision with 2^d -Trees

Historically:

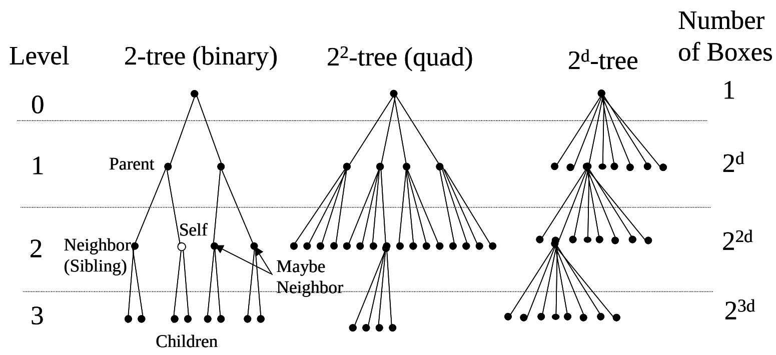
- Binary trees (1D), Quadtrees (2D), Octrees (3D);
- We will consider a concept of 2^d-tree.
  - d=1 – binary;
  - d=2 – quadtree;
  - d=3 – octree;
  - d=4 – hexatree;
  - and so on..

Hierarchy in 2^d-tree



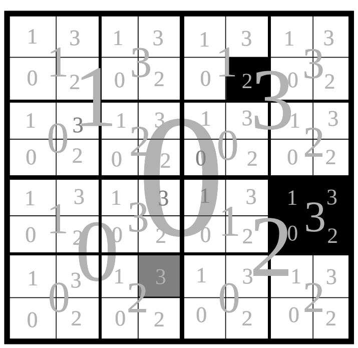
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

2^d-trees



Hierarchical Indexing

Hierarchical Indexing in 2^d-trees. Index at the Level.



Indexing in quad-tree

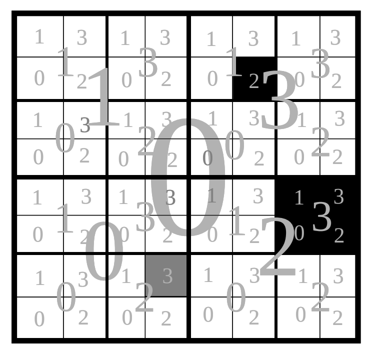
The large black box has the indexing string (2,3). So its index is $23_4=11_{10}$.

The small black box has the indexing string (3,1,2). So its index is $312_4=54_{10}$.

In general: Index (Number) at level l is:

$$Number = (2^d)^{l-1} \cdot N_1 + (2^d)^{l-2} \cdot N_2 + \ldots + 2^d \cdot N_{l-1} + N_l.$$

Universal Index (Number)



The large black box has the indexing string (2,3). So its index is $23_4=11_{10}$ at level 2

The small gray box has the indexing string (0,2,3). So its index is $23_4=11_{10}$ at level 3.

In general: Universal index is a pair:

$$UniversalNumber = (Number, l)$$

This index at this level

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

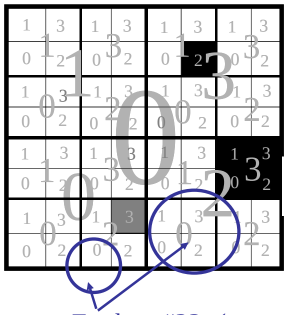
Parent Index

Parent's indexing string:

$$Parent(N_1, N_2, \ldots, N_{l-1}, N_l) = (N_1, N_2, \ldots, N_{l-1}).$$

Parent's index:

$$Parent(Number) = \left(2^d\right)^{l-2} \bullet N_1 + \left(2^d\right)^{l-3} \bullet N_2 + \ldots + N_{l-1}.$$



Parent index does not depend on the level of the box! E.g. in the quad-tree at any level

$$Parent(11_{10}) = Parent(23_4) = 2_4 = 2_{10}.$$

Parent's universal index:

$$Parent((Number, l)) = (Parent(Number), l-1).$$

Algorithm to find the parent number:

$$Parent(Number) = \left[Number/2^d\right]$$

For box #23, (gray or black) the parent box index is 2.

Children Indexes

Children indexing strings:

$$Children(N_1, N_2, \ldots, N_{l-1}, N_l) = \{(N_1, N_2, \ldots, N_{l-1}, N_l, N_{l+1})\}, \quad N_{l+1} = 0, \ldots, 2^d - 1.$$

Children indexes:

$$Children(Number) = \left\{\left(2^d\right)^{l} \bullet N_1 + \left(2^d\right)^{l-1} \bullet N_2 + \ldots + \left(2^d\right) \bullet N_l + N_{l+1}\right\}, \quad N_{l+1} = 0, \ldots, 2^d - 1.$$

Children indexes do not depend on the level of the box! E.g. in the quad-tree at any level:

$$Children(11_{10}) = Children(23_4) = \{230_4, 231_4, 232_4, 233_4\} = \{44_{10}, 45_{10}, 46_{10}, 47_{10}\}$$

Children universal indexes:

$$Children((Number, l)) = (Children(Number), l+1).$$

Algorithm to find the children numbers:

$$Children(Number) = \left\{2^d \bullet Number + j\right\}, \quad j = 0, \ldots, 2^d - 1,$$

A couple of examples:

Problem: Using the above numbering system and decimal numbers find parent box number for box #5981 in oct-tree.
Solution: Find the integer part of division of this number by 8. [5981/8] = 747.
Answer: #747.

Problem: Using the above numbering system and decimal numbers find children box numbers for box #100 in oct-tree.
Solution: Multiply this number by 8 and add numbers from 0 to 7.
Answer: ##800, 801, 802, 803, 804, 805, 806, 807.

Can it be even faster?

YES!
USE BITSHIFT PROCEDURES!

(HINT: Multiplication and division by 2^d are equivalent to d-bit shift.)

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Matlab Program for Parent Finding

Box Index

```
p = bitshift(n,-d);
```

Parent Box Index

Space dimensionality

Binary Ordering

All $\bar{x} \in [0,1]$ naturally ordered and can be represented in decimal system as

$$\bar{x} = (0.a_1 a_2 a_3 ...)_{10}, \quad a_j = 0,...,9; \quad j = 1,2,...$$

Note that the point $\bar{x} = 1$ can be written not only $\bar{x} = 1.0000...$, but also as

$$\bar{x} = 1 = (0.999999....)_{10}$$

We also can represent any point $\bar{x} \in [0,1]$ in binary system as

$$\bar{x} = (0.b_1 b_2 b_3 ...)_2, \quad b_j = 0,1; \quad j = 1,2,...$$

By the same reasons as for decimal system the point $\bar{x} = 1$ can be written as

$$\bar{x} = 1 = (0.111111....)_2.$$

Finding the index of the box containing a given point

Level	Box Size (dec)	Box Size (bin)
0	1	1
1	0.5	0.1
2	0.25	0.01
3	0.125	0.001
...	...	...

Level 1:

$$(0.0b_1b_2b_3...)_2 \in Box((0)), \quad (0.1b_1b_2b_3...)_2 \in Box((1)), \quad \forall b_j = 0,1; \quad j = 1,2,...,$$

Level 2:

$$(0.00b_1b_2b_3...)_2 \in Box((0,0)), \quad (0.01b_1b_2b_3...)_2 \in Box((0,1)),$$
$$(0.10b_1b_2b_3...)_2 \in Box((1,0)), \quad (0.11b_1b_2b_3...)_2 \in Box((1,1)),$$
$$\forall b_j = 0,1; \quad j = 1,2,...,$$

Level l:

$$(0.N_1N_2...N_lb_1b_2b_3...)_2 \in Box((N_1,N_2,...,N_l)), \quad \forall b_j = 0,1; \quad j = 1,2,...$$

We use indexing strings !

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

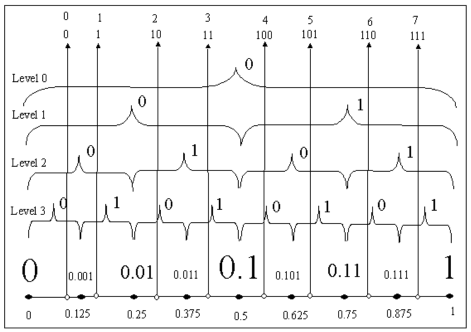
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Finding the index of the box containing a given point (2)

$$(0.N_1N_2...N_lb_1b_2b_3...)_2 \to (N_1N_2...N_l.b_1b_2b_3...)_2; \quad N_1N_2...N_l = [(N_1N_2...N_l.b_1b_2b_3...)_2].$$

$$(Number, l) = [2^l \cdot \bar{x}].$$

This is an algorithm for finding of the box index at level l (!).



Faster method: Use bit shift and take prefix.

Finding the center of a given box.

For box number *Number* at level l the left boundary can be found by l –bit shift:

$$Number = (N_1N_2...N_l)_2 \to (0.N_1N_2...N_l)_2,$$

Add 1 as an extra digit (half of the box size), so we have for the center of the box at level l :

$$\bar{x}_c(Number, l) = (0.N_1N_2...N_l1)_2.$$

This procedure also can be written in the form that does not depend on the counting system:

$$\bar{x}_c(Number, l) = 2^{-l} \cdot Number + 2^{-l-1} = 2^{-l} \cdot (Number + 2^{-1}).$$

since addition of one at position $l+1$ after the point in the binary system is the same as addition of 2^{-l-1}.

Problem: Find the center of box #31 (decimal) at level 5 of the binary tree.

Solution: We have $\bar{x}_c(31, 5) = 2^{-5} \cdot (31 + 0.5) = 0.984375$.

Answer: 0.984375.

This is the algorithm!

Neighbor finding

Due to all boxes are indexed consequently:

Neighbor((Number,level))=Number±1

If the neighbor number at level l equal 2^l or -1 we drop this box from the neighbor list.

Problem: Find all neighbors of box #31 (decimal) at level 5 of the binary tree.

Solution: The neighbors should have numbers $31 - 1 = 30$ and $31 + 1 = 32$. However, $32 = 2^5$, which exceeds the number allowed for this level. Thus, only box #30 is the neighbor.

Answer: #30.

Spatial Ordering Using Bit Interleaving

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Bit Interleaving

Coordinates of a point $\bar{\mathbf{x}} = (\bar{x}_1,...,\bar{x}_d)$ in the d-dimensional unit cube can be represented in binary form

$$\bar{x}_k = (0.b_{k1}b_{k2}b_{k3}...)_2, \quad b_{kj} = 0,1; \quad j = 1,2,..., \quad k = 1,...,d.$$

Instead of having d numbers characterizing each point we can form a single binary number that represent the same point by ordered mixing of the digits in the above binary representation (this is also called *bit interleaving*), so we can write:

$$\bar{\mathbf{x}} = (0.b_{11}b_{21}...b_{d1}b_{12}b_{22}...b_{d2}...b_{1j}b_{2j}...b_{dj}...)_2.$$

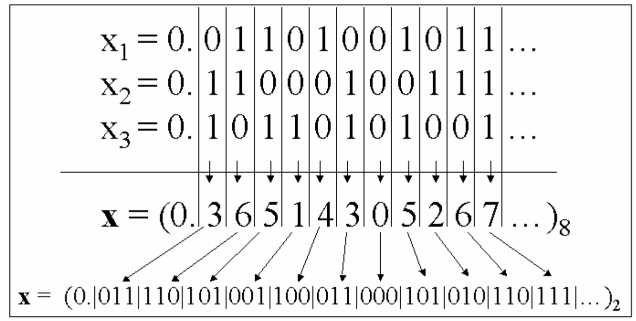
This number can be rewritten in the system with base 2^d:

$$\bar{\mathbf{x}} = (0.N_1N_2N_3...N_j...)_{2^d}, \quad N_j = (b_{1j}b_{2j}...b_{dj})_2, \quad j = 1,2,..., \quad N_j = 0,...,2^d-1.$$

This maps $\mathbf{R}^d \to \mathbf{R}$, where coordinates are ordered naturally!

Example of Bit Interleaving.

Consider 3-dimensional space, and an octree.

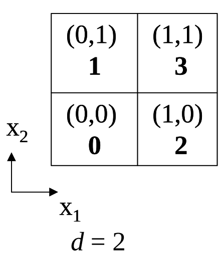


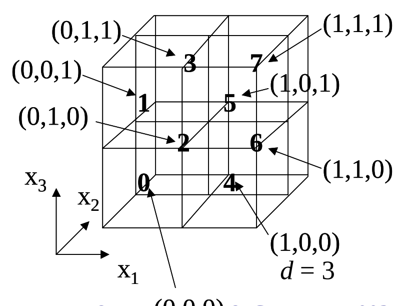
Convention for Children Ordering.

Any binary string of length d can be converted into a a single number (binary or in some other counting system, e.g. with the base 2^d):

$$(b_1,b_2,...,b_d) \to (b_1b_2...b_d)_2 = N_{2^d}.$$

This provides natural numbering of 2^d children of the box.`





Finding the index of the box containing a given point.

Level 1:

$$\bar{\mathbf{x}} = (0.b_{11}b_{21}...b_{d1}b_{12}b_{22}...b_{d2}...b_{1j}b_{2j}...b_{dj}...)_2 \in Box((b_{11}b_{21}...b_{d1})_2) = Box((N_1)_{2^d}),$$

Let us use 2^d-based counting system. Then we can find the box containing a given point at Level l :

$$(0.N_1N_2...N_lc_1c_2c_3...)_{2^d} \in Box((N_1,N_2,...,N_l)_{2^d}), \quad \forall c_j = 0,...,2^d-1; \quad j = 1,2,...$$

Therefore to find the number of the box at level l to which the given point belongs we need simply shift the 2^d number representing this point by l positions and take the integer part of this number:

$$(0.N_1N_2...N_lc_1c_2c_3...)_{2^d} \to (N_1N_2...N_l.c_1c_2c_3...)_{2^d}; \quad N_1N_2...N_l = [(N_1N_2...N_l.b_1b_2b_3...)_{2^d}].$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Finding the index of the box containing a given point. Algorithm and Example.

This procedure also can be performed in binary system by $d \cdot l$ bit shift:

$$(0.b_{11}b_{21}...b_{d1}b_{12}b_{22}...b_{d2}...b_{1l}b_{2l}...b_{dl}b...)_2 \to (b_{11}b_{21}...b_{d1}b_{12}b_{22}...b_{d2}...b_{1l}b_{2l}...b_{dl}.b...)_2;$$

$$Number = (b_{11}b_{21}...b_{d1}b_{12}b_{22}...b_{d2}...b_{1l}b_{2l}...b_{dl})_2.$$

In arbitrary counting system:

$$(Number, l) = \left[2^{dl} \cdot \bar{\mathbf{x}}\right].$$

Problem: Find decimal numbers of boxes at levels 3 and 5 of the oct-tree containing point $\bar{\mathbf{x}} = (0.7681, 0.0459, 0.3912)$.

Solution: First we convert the coordinates of the point to binary format, where we can keep only 5 digits after the point (maximum level is 5), so $\bar{\mathbf{x}} = (0.11000, 0.00001, 0.01100)_2$. Second, we form a single mixed number $\bar{\mathbf{x}} = 0.100101001000010_2$. Performing $3 \cdot 3 = 9$ bit shift and taking integer part we have $(Number, 3) = 100101001_2 = 297$. Performing $3 \cdot 5 = 15$ bit shift we obtain $(Number, 5) = 100101001000010_2 = 19010$.

Answer: #297 and #19010.

Bit Deinterleaving

Convert the box number at level l into binary form

$$Number = (b_{11}b_{21}...b_{d1}b_{12}b_{22}...b_{d2}...b_{1l}b_{2l}...b_{dl})_2.$$

Then we decompose this number to d numbers that will represent d coordinates:

$$Number_1 = (b_{11}b_{12}...b_{1l})_2.$$

$$Number_2 = (b_{21}b_{22}...b_{2l})_2.$$

...

$$Number_d = (b_{d1}b_{d2}...b_{dl})_2.$$

Bit deinterleving (2). Example.

Number = 76893_{10}

It is OK that the first group is incomplete

To break the number into groups of d bits start from the last digit!

10010110001011101_2

$Number_3$	=	0	0	0	1	1	1	=	111_2	=	7_{10}
$Number_2$	=	1 1	1 1	0	1	0	=	111010_2	=	58_{10}
$Number_1$	=	0	1	0	0	1		=	1001_2	=	9_{10}

Finding the center of a given box.

Coordinates of the box center in binary form are

$$\bar{x}_{k,c}(Number, l) = (0.b_{k1}b_{k2}...b_{kl}1)_2, \quad k = 1,...,d.$$

or in the form that does not depend on the counting system:

$$\bar{x}_{k,c}(Number, l) = 2^{-l} \cdot Number_k + 2^{-l-1} = 2^{-l} \cdot \left(Number_k + \frac{1}{2}\right), \quad k = 1,...,d.$$

Problem: Find the center of box #533 (decimal) at level 5 of the oct-tree.

Solution: Converting this number to the bit string we have $533_{10} = 1000010101_2$. Retrieving the digits of three components from the last digit of this number we obtain: $Number_3 = 1001_2 = 9_{10}$, $Number_2 = 10_2 = 2_{10}$, $Number_1 = 1_2 = 1_{10}$. We have then $\bar{x}_{1,c}(533,5) = 2^{-5} \cdot (1 + 0.5) = 0.04875$, $\bar{x}_{2,c}(533,5) = 2^{-5} \cdot (2 + 0.5) = 0.078125$, $\bar{x}_{3,c}(533,5) = 2^{-5} \cdot (9 + 0.5) = 0.296875$.

Answer: $\bar{\mathbf{x}}_c = (0.04875, 0.078125, 0.296875)$.

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Neighbor Finding

Step 1: Deinterleaving:

$$Number \to \{Number_1, \ldots, Number_d\}$$

Step 2: Shift of the coordinate numbers

$$Number_k^+ = Number_k + 1, \quad Number_k^- = Number_k - 1, \quad k = 1, \ldots, d,$$

and formation of sets:

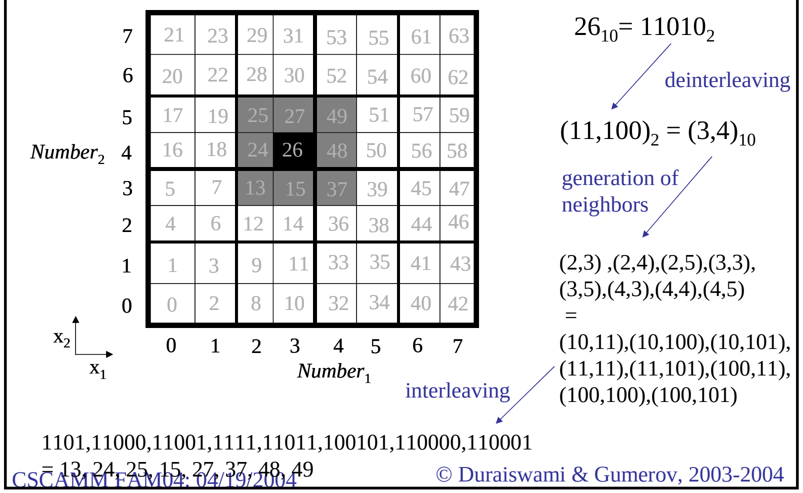
$$s_k = \begin{cases} \{Number_k^-, Number_k, Number_k^+\}, & Number_k \neq 0, 2^l - 1 \\ \{Number_k, Number_k^+\}, & Number_k = 0. \\ \{Number_k^-, Number_k\}, & Number_k = 2^l - 1. \end{cases} \qquad k = 1, \ldots, d.$$

The set of neighbor generating numbers is then

$$n = (n_1, \ldots, n_d), \quad n_k \in s_k, \quad k = 1, \ldots, d.$$

where each n_k can be any element of s_k, except of the case when all $n_k = Number_k$ simultaneously for all $k = 1, \ldots, d,$ since this case corresponds to the box itself.

Example of Neighbor Finding



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Spatial Data Structuring

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Definitions

Definitions

Data Collection.

Scaling and mapping finite d-dimensional data into a unit d-dimensional cube yields in a *collection* C of N points distributed inside such a cube:

$$\mathrm{C} = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_N\}, \quad \bar{\mathbf{x}}_i \in [0,1) \times [0,1) \times \ldots \times [0,1) \subset \mathrm{R}^d, \quad i = 1, \ldots, N.$$

Data Set.

We call a collection $\mathrm{C} = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_N\}$ "*data set*", if $\forall i \neq j$, $dist(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) \neq 0$, where $dist(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$ denotes distance between $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}_j$.

Non-Separable (Multi-entry) Data Collection.

We call a collection $\mathrm{C} = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_N\}$ "*non-separable data collection*", if $\exists i \neq j$, $dist(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = 0$.

(By this definition a non-separable data collection cannot be uniquely ordered using distance function $dist(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$).

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Threshold Level

We call level $L_{th}(C)$ "*threshold level*" of data collection C if the maximum number of data points in a box for any level of subdivision $L > L_{th}(C)$ is the same as for $L_{th}(C)$ and differs from $L_{th}(C)$ for any $L < L_{th}(C)$.

Note: in case if C is a data set of power $N \geqslant 2$, then at level $L_{th}(C)$ we will have maximum one data point per box, and at $L < L_{th}(C)$ there exists at least 1 box containing 2 or more data points.

Spatial Data Sorting

Consider data collection C. Each point can be then indexed (or numbered):

$$\mathbf{v} = (v_1, v_2, \ldots, v_N), \quad v_i = Number(\mathbf{x}_i, L), \quad i = 1, \ldots, N,$$

where *Number* can be determined using the algorithm described in the previous sections.
The array v then can be sorted for $O(N \log N)$ operations:

$$(v_1, v_2, \ldots, v_N) \to (v_{i_1}, v_{i_2}, \ldots, v_{i_N}), \quad v_{i_1} \leqslant v_{i_2} \leqslant \ldots \leqslant v_{i_N}.$$

using standart sorting algorithms. These algorithms also return the permutation index (other terminology can be permutation vector or pointer vector) of length N :

$$\mathbf{ind} = (i_1, i_2, \ldots, i_N),$$

that can be stored in the memory. In terms of memory usage the array v should not be rewritten and stored again, since ind is a pointer and

$$\mathbf{v}(i) = v_i, \quad \mathbf{ind}(j) = i_j, \quad \mathbf{v}(\mathbf{ind}(j)) = \mathbf{v}(i_j) = v_{i_j}, \quad i,j = 1, \ldots, N,$$

so

$$\mathbf{v}(\mathbf{ind}) = (v_{i_1}, v_{i_2}, \ldots, v_{i_N}).$$

Spatial Data Sorting (2)

- Before sorting represent your data with maximum number of bits available (or intended to use). This corresponds to maximum level $L_{available}$ available (say [$L_{available} = BitMax/d$].
- In the hierarchical 2^d-tree space subdivision the sorted list will remain sorted at any level $L < L_{available}$. So the data ordering is required only one time.

After data sorting we need to find the maximum level of space subdivision that will be employed

In Multilevel FMM two following conditions can be mainly considered:

- At level L_{max} each box contains not more than s points (s is called clustering or grouping parameter)
- At level L_{max} the neighborhood of each box contains not more than q points.

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

The threshold level determination algorithm in $O(N)$ time

$i = 0,\ m = s,$
while $m < N$
  $i = i + 1,\ m = m + 1;$
  $a = Interleaved(v(\mathbf{ind}(i)));$
  $b = Interleaved(v(\mathbf{ind}(m)));$
  $j = Bit_{max} + 1$
  while $a \neq b$
    $j = j - 1;$
    $a = Parent(a);$
    $b = Parent(b);$
    $l_{max} = \max(l_{max}, j);$
  end;
end;

s is the clustering parameter

Binary Search in Sorted List

- Operation of getting non-empty boxes at any level L (say neighbors) can be performed with O(log$N$) complexity for any fixed d.
  - It consists of obtaining a small list of all neighbor boxes with O(1) complexity and
  - Binary search of each neighbor in the sorted list at level L is an $O(Ld)$ operation.
  - For small L *and* d this is almost $O(1)$ procedure.

Operations on Sets

Difference: $\mathbb{C} = \mathbb{A} \backslash \mathbb{B}$
Intersection: $\mathbb{C} = \mathbb{A} \cap \mathbb{B}$
Union: $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$

Let $Pow(\mathbb{A}) = N,\quad Pow(\mathbb{B}) = M,\quad N \geqslant M,$

Then the complexity for sorted input/output:

$$\mathbb{A} \backslash \mathbb{B} : N$$
$$\mathbb{A} \cap \mathbb{B} : \min(N, M \log N)$$
$$\mathbb{A} \cup \mathbb{B} : N$$

Operations

$$Neighbors(\mathbb{W}; n, l) = NeighborsAll(n, l) \cap \mathbb{W}, \quad \mathbb{W} = \mathbb{X}, \mathbb{Y},$$
$$Children(\mathbb{W}; n, l) = ChildrenAll(n, l) \cap \mathbb{W}, \quad \mathbb{W} = \mathbb{X}, \mathbb{Y}.$$

are $O(\log N)$ operations for minimum memory requirements and $O(1)$ for sufficiently large memory.

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

The Multilevel Fast Multipole Method

Ramani Duraiswami

Nail Gumerov

Review

$$\mathbf{v} = \mathbf{\Phi u},$$

- FMM aims at accelerating the matrix vector product
- Matrix entries determined by a set of source points and evaluation points (possibly the same)

$$\mathbf{\Phi} = \begin{pmatrix} \Phi(\mathbf{y}_1,\mathbf{x}_1) & \Phi(\mathbf{y}_1,\mathbf{x}_2) & \dots & \Phi(\mathbf{y}_1,\mathbf{x}_N) \\ \Phi(\mathbf{y}_2,\mathbf{x}_1) & \Phi(\mathbf{y}_2,\mathbf{x}_2) & \dots & \Phi(\mathbf{y}_2,\mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ \Phi(\mathbf{y}_M,\mathbf{x}_1) & \Phi(\mathbf{y}_M,\mathbf{x}_2) & \dots & \Phi(\mathbf{y}_M,\mathbf{x}_N) \end{pmatrix}.$$

$$\mathsf{X} = \{\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N\}, \quad \mathbf{x}_i \in \mathsf{R}^d, \quad i = 1,\dots,N,$$
$$\mathsf{Y} = \{\mathbf{y}_1,\mathbf{y}_2,\dots,\mathbf{y}_M\}, \quad \mathbf{y}_j \in \mathsf{R}^d, \quad j = 1,\dots,M.$$

$$v_j = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}_j,\mathbf{x}_i), \quad j = 1,\dots,M.$$

- Function Φ has following point-centered representations about a given point x_*
  - Local (valid in a neighborhood of a given point)
  - Far-field or multipole (valid outside a neighborhood of a given point)
  - In many applications Φ is singular
- Representations are usually series
  - Could be integral transform representations
- Representations are usually approximate
  - Error bound guarantees the error is below a specified tolerance

Review

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j) + Error(p,\mathbf{x}_i,\mathbf{y}_j).$$

- One representation, valid in a given domain, can be converted to another valid in a subdomain contained in the original domain
- Factorization trick is at core of the FMM speed up

$$v_j = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{i=1}^{N} u_i \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j) + \sum_{i=1}^{N} u_i Error(p,\mathbf{x}_i,\mathbf{y}_j)$$
$$= \sum_{m=0}^{p-1} B_m F_m(\mathbf{y}_j) + Error_j(p,N), \quad j = 1,\dots,M.$$

- Representations we use are factored … separate points x_i and y_j
- Data is partitioned to organize the source points and evaluation points so that for each point we can separate the points over which we can use the factorization trick, and those we cannot.
- Hierarchical partitioning allows use of different factorizations for different groups of points
- Accomplished via MLFMM

Prepare Data Structures

- Convert data set into integers given some maximum number of bits allowed/dimensionality of space
- Interleave
- Sort
- Go through the list and check at what bit position two strings differ
  - For a given *s* determine the number of levels of subdivision needed
  - *s* is the maximum number of points in a box at the finest level

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

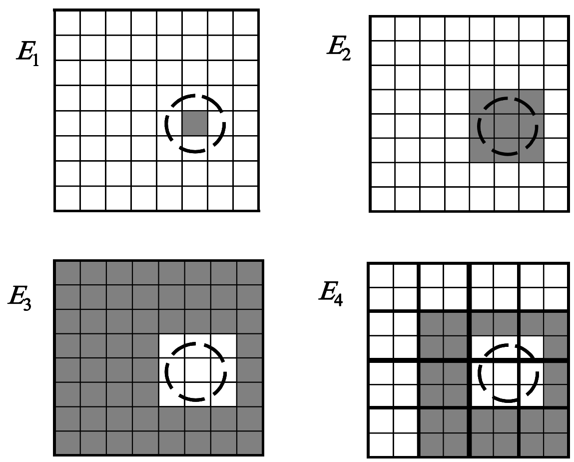
Hierarchical Spatial Domains

E_1: box

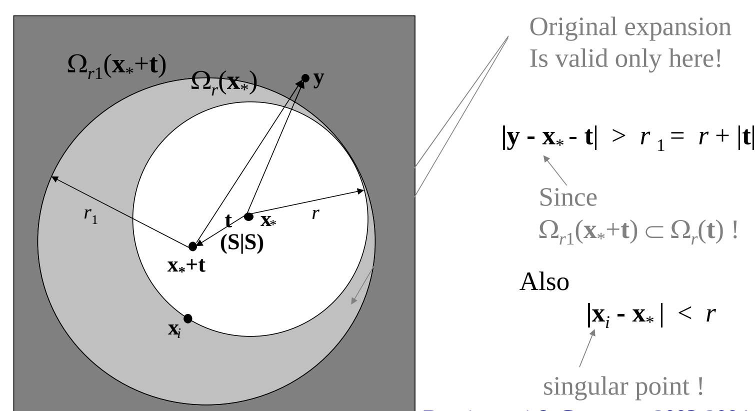
E_2: points in the box and in neighboring boxes

E_3: points in boxes outside neighborhood

E_4: points belonging to neighbors of parent box, but which do not belong to E_2



S|S-reexpansion (Far to Far, or Multipole to Multipole, or M2M)



UPWARD PASS

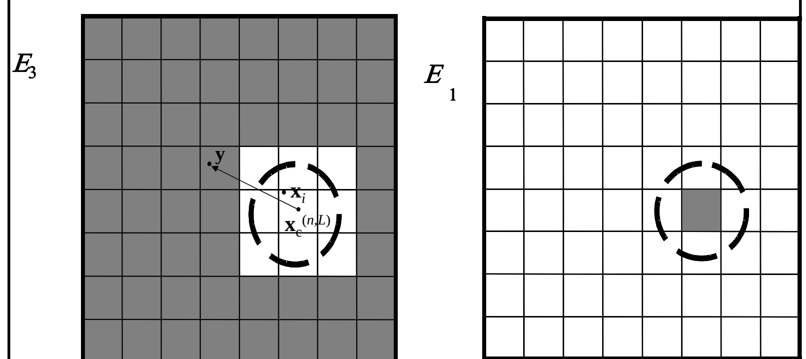
- Partition sources into a source hierarchy.
- Stop hierarchy so that boxes at the finest level contain at most *s* sources
- Let the number of levels be *L*
- Consider the finest level
- For non-empty boxes we create *S* expansion about center of the box $\Phi(x_i,y)=\Sigma^P u_i B(x_*,x_i)\, S(x_*,y)$

$$\Phi_1^{(n,L)}(\mathbf{y}) = \mathbf{C}^{(n,L)} \circ \mathbf{S}(\mathbf{y}-\mathbf{x}_c^{(n,L)}),$$

$$\mathbf{C}^{(n,L)} = \sum_{\mathbf{x}_i \in E_1(n,L)} u_i \mathbf{B}\left(\mathbf{x}_i, \mathbf{x}_c^{(n,L)}\right).$$

- We need to keep these coefficients. $\boldsymbol{C}^{(n,l)}$ for each level as we will need it in the downward pass
- Then use S/S translations to go up level by level up to level 2.
- Cannot go to level 1 (Why?)

- *S* expansion is valid in the domain *E_3* outside domain *E_1* (provided *d*<9)



Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

UPWARD PASS

- At the end of the upward pass we have a set of *S* expansions (i.e. we have coefficients for them)
- we have a set of coefficients $C^{(n,l)}$ for $n=1,\ldots,2^{ld}$ $l=L,\ldots,2$
- Each of these expansions is about a center, and is valid in some domain
- We would like to use the coarsest expansions in the downward pass (have to deal with fewest numbers of coefficients)
- But may not be able to --- because of domain of validity
- Upward pass works on source points and builds representations to be used in the downward pass, where the actual product will be evaluated

DOWNWARD PASS

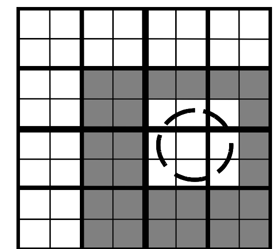
- Starting from level 2, build an *R* expansion in boxes where *R* expansion is valid

$$\Phi_4^{(n,l)}(\mathbf{y}) = \widetilde{\mathbf{D}}^{(n,l)} \circ \mathbf{R}(\mathbf{y} - \mathbf{x}_c^{(n,l)}),$$

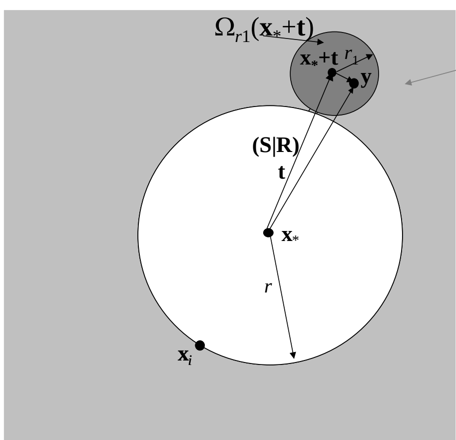
$$\widetilde{\mathbf{D}}^{(n,l)} = \sum_{m \in I_4(n,l)} (\mathbf{S}|\mathbf{R})\left(\mathbf{x}_c^{(n,l)} - \mathbf{x}_c^{(m,l)}\right)\mathbf{C}^{(m,l)}.$$

- Must to do *S*|*R* translation
- The *S* expansion is not valid in boxes immediately surrounding the current box
- So we must exclude boxes in the E_4 neighborhood

E_4



S|R-reexpansion (Far to Local, or Multipole to Local, or M2L)



Original expansion
Is valid only here!

$|\mathbf{y} - \mathbf{x}_* - \mathbf{t}| < r_1 = |\mathbf{t}| - r$

Since
$\Omega_{r1}(\mathbf{x}_*+\mathbf{t}) \subset \Omega_r(\mathbf{t})$!

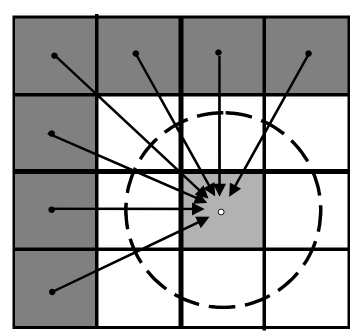
Also

$|\mathbf{x}_i - \mathbf{x}_*| < r$

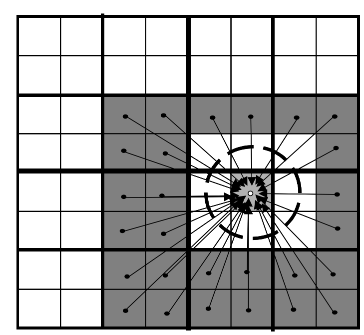
singular point !

Downward Pass. Step 1.

Level 2:



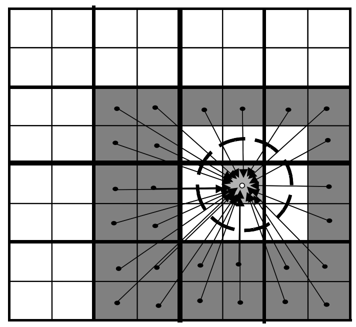
Level 3:



gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

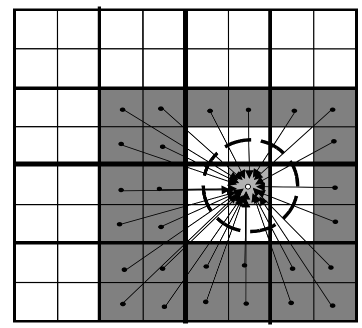
Downward Pass. Step 1.

THIS MIGHT BE
THE MOST EXPENSIVE
STEP OF THE ALGORITHM



Downward Pass. Step 1.

$$P_4 = PowerOfE_4Neighborhood = 3^d 2^d - 3^d = 3^d (2^d - 1)$$



$d = 1: \quad P_4 = 3,$

$d = 2: \quad P_4 = 27,$

$d = 3: \quad P_4 = 189,$

$d = 4: \quad P_4 = 1215,$

...

Exponential Growth

Total number of S|R-translations per 1 box in d-dimensional space

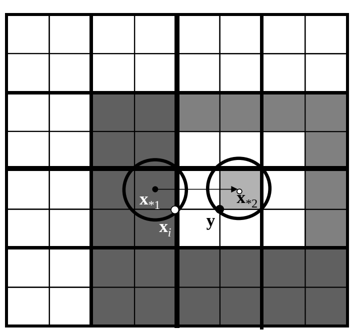
(far from the domain boundaries)

Domains of Expansion Validity (6). S|R-translation.

S-expansion coefficients can be $S|R$-translated (converted to R-expansion coefficients)

$$|\mathbf{y} - \mathbf{x}_{*2}| < |\mathbf{x}_{*1} - \mathbf{x}_{*2}| - |\mathbf{x}_i - \mathbf{x}_{*1}|,$$

$$\mathrm{A}(\mathbf{x}_i, \mathbf{x}_{*2}) = (\mathbf{S}|\mathbf{R})(\mathbf{x}_{*2} - \mathbf{x}_{*1})\mathrm{B}(\mathbf{x}_i, \mathbf{x}_{*1})$$



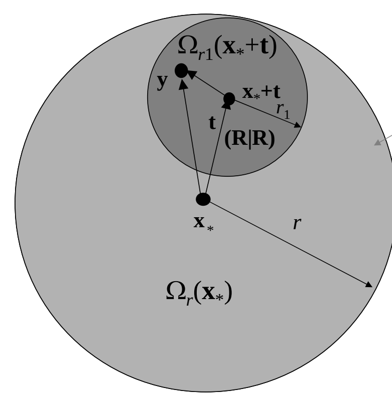
$$|\mathbf{y} - \mathbf{x}_*| < r_{\min}(l), \quad |\mathbf{x}_i - \mathbf{x}_*| < r_{\min}(l),$$

$$\min|\mathbf{x}_{*1} - \mathbf{x}_{*2}| = 2size(l).$$

$$2^{-l-1}\sqrt{d} + 2^{-l-1}\sqrt{d} < 2^{-l+1}, \quad d < 4.$$

$d < 4$

R|R-reexpansion (Local to Local, or L2L)



Original expansion
Is valid only here!

$|\mathbf{y} - \mathbf{x}_* - \mathbf{t}| < r_1 = r - |\mathbf{t}|$

Since $\Omega_{r1}(\mathbf{x}_*+\mathbf{t}) \subset \Omega_r(\mathbf{t})$!

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Downward Pass Step 2

- Now consider we already have done the S|R translation at some level at the center of a box.
- So we have a R expansion that includes contribution of most of the points, but not of points in the E_4 neighborhood
- We can go to a finer level to include these missed points
- But we will now have to translate the already built R expansion to a box center of a child
  - (Makes no sense to do S|R again, since many S|R are consolidated in this R expansion)
- Add to this translated one, the S|R of the E_4 of the finer level

- Formally

Step 2. At $l = 2$ we have

$$\Phi_3^{(n,2)}(\mathbf{y}) = \Phi_4^{(n,2)}(\mathbf{y}), \quad \mathbf{D}^{(n,2)} = \widetilde{\mathbf{D}}^{(n,2)},$$

Form $\Phi_3^{(n,l)}(\mathbf{y})$ (or expansion coefficients of this function) by adding $\Phi_4^{(Parent(n),l-1)}(\mathbf{y})$ to (R|R)- translated coefficients of the parent box to the child center:

$$\Phi_3^{(n,l)}(\mathbf{y}) = \mathbf{D}^{(n,l)} \circ \mathbf{R}(\mathbf{y} - \mathbf{x}_c^{(n,l)}),$$

$$\mathbf{D}^{(n,l)} = \widetilde{\mathbf{D}}^{(n,l)} + (\mathbf{R}|\mathbf{R})\left(\mathbf{x}_c^{(n,l)} - \mathbf{x}_c^{(m,l-1)}\right)\mathbf{D}^{(m,l-1)}, \quad m = Parent(n).$$

$$\Phi_4^{(n,l)}(\mathbf{y}) = \widetilde{\mathbf{D}}^{(n,l)} \circ \mathbf{R}(\mathbf{y} - \mathbf{x}_c^{(n,l)}),$$

$$\widetilde{\mathbf{D}}^{(n,l)} = \sum_{m \in I_4(n,l)} (\mathbf{S}|\mathbf{R})\left(\mathbf{x}_c^{(n,l)} - \mathbf{x}_c^{(m,l)}\right)\mathbf{C}^{(m,l)}.$$

Downward Pass. Step 2.

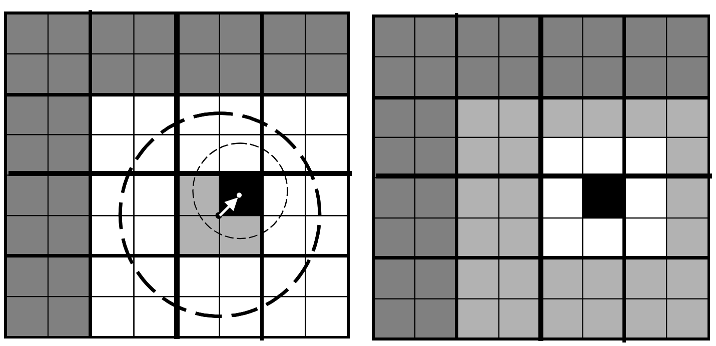


Figure shows that local-to-local translation is applicable in this case (smaller sphere is located completely inside the larger sphere), and junction of structures $E_3(n,l)$ and $E_4(n,l+1)$ produces $E_3(n,l+1)$:

$$E_3(n,l+1) = E_3(n,l) \cup E_4(n,l+1).$$

Domains of Expansion Validity (5). R|R and S|S-translations.

- *R*-expansion coefficients can be *R|R*-translated:

$$|\mathbf{y} - \mathbf{x}_{*2}| < |\mathbf{x}_i - \mathbf{x}_{*1}| - |\mathbf{x}_{*1} - \mathbf{x}_{*2}| :$$

$$\mathbf{A}(\mathbf{x}_i, \mathbf{x}_{*2}) = (\mathbf{R}|\mathbf{R})(\mathbf{x}_{*2} - \mathbf{x}_{*1})\mathbf{A}(\mathbf{x}_i, \mathbf{x}_{*1})$$

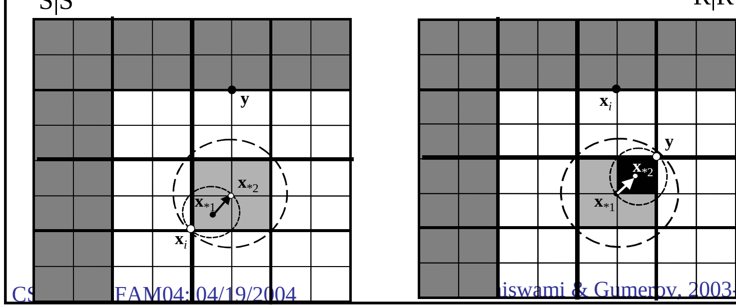
- *S*-expansion coefficients can be *S|S*-translated:

$$|\mathbf{y} - \mathbf{x}_{*2}| > |\mathbf{x}_{*1} - \mathbf{x}_{*2}| + |\mathbf{x}_i - \mathbf{x}_{*1}|,$$

$$\mathbf{B}(\mathbf{x}_i, \mathbf{x}_{*2}) = (\mathbf{S}|\mathbf{S})(\mathbf{x}_{*2} - \mathbf{x}_{*1})\mathbf{B}(\mathbf{x}_i, \mathbf{x}_{*1})$$

Not as restrictive as S|R

S|S R|R



Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Final Summation

- At this point we are at the finest level.
- We cannot do any S|R translation for x_i 's that are in the E_3 neighborhood of our y_j's
- Must evaluate these directly

CSCAMM FAM04: 04/19/2004

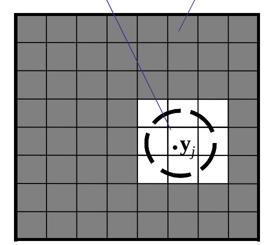
Final Summation

As soon as coefficients $\mathbf{D}^{(n,L)}$ are determined total potential can be computed for any point $\mathbf{y}_j \in E_1(0,0)$, where $\Phi_2^{(n,l)}(\mathbf{y})$ can be computed straightforward. So:

$$v_j = \Phi(\mathbf{y}_j) = \sum_{\mathbf{x}_i \in E_2(n,L)} u_i \Phi(\mathbf{y}_j, \mathbf{x}_i) + \mathbf{D}^{(n,L)} \circ \mathbf{R}(\mathbf{y}_j - \mathbf{x}_c^{(n,L)}), \quad \mathbf{y}_j \in E_1(n,L).$$

Contribution of E_2

Contribution of E_3



CSCAMM FAM04: 04/19/2004

Cost of FMM --- Upward Pass

- Upward Step1. Cost of creating an S expansion for each source point. *O(NP)*
- Upward Step2. Cost of performing an S|S translation
  - If we use expensive (matrix vector) method cost is $O(P^2)$ for one translation.
- Step 2 is repeated from level *L-1* to level *2*

$$CostUpward_2 = 2^d \left(2^{(L-1)d} + 2^{(L-2)d} + \ldots + 2^{2d}\right) CostSS(P)$$
$$< \frac{2^d}{2^d - 1}\left(2^{Ld} - 1\right) CostSS(P) \sim \frac{N}{s} CostSS(P)$$

- Total Cost of Upward Pass $\sim NP + (N/s)(P^2)$

CSCAMM FAM04: 04/19/2004

COST of MLFMM

- Cost of downward pass, step 1 is the cost of performing S|R translations at each level

$$CostDownward_1 \lesssim P_4(d)\left(2^{2d} + \ldots + 2^{Ld}\right) CostSR(P) \sim P_4(d)\frac{N}{s} CostSR(P),$$

- At the downward pass, 2nd step we have the cost of the R|R translation, and S|R translation from the E_4 neighbourhood (already accounted for above)

$$CostDownward_2 = 2^d\left(2^{2d} + \ldots + 2^{(L-1)d}\right) CostRR(P) \sim \frac{N}{s} CostRR(P),$$

- Final summation cost is $CostEvaluation = M(P_2(d) sCostFunc + P).$

- Total

$$CostMLFMM = (M+N)P + (P_4(d)+2)\frac{N}{s} CostTranslation(P) + P_2(d) sMCostFunc$$

CSCAMM

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Itemized Cost of MLFMM

Regular mesh:

$N = 2^{L_* d}, \quad s = 2^{L_s d}, \quad L = L_{\max} = L_* - L_s.$

Assume that all translation costs are the same, *CostTranslation*(*P*)

$$CostUpward_1 = NCostExpansion(P) = O(NP).$$

$$CostUpward_2 = 2^d\left(2^{(L-1)d} + 2^{(L-2)d} + ... + 2^{2d}\right)CostSS(P)$$

$$< \frac{2^d}{2^d - 1}\left(2^{Ld} - 1\right)CostSS(P) \sim \frac{N}{s}CostSS(P)$$

$$CostDownward_1 \lesssim P_4(d)\left(2^{2d} + ... + 2^{Ld}\right)CostSR(P) \sim P_4(d)\frac{N}{s}CostSR(P),$$

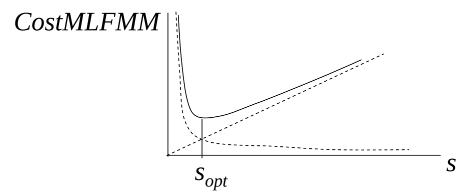
$$CostDownward_2 = 2^d\left(2^{2d} + ... + 2^{(L-1)d}\right)CostRR(P) \sim \frac{N}{s}CostRR(P),$$

$$CostEvaluation = M(P_2(d)sCostFunc + P).$$

Powers of E_4 and E_2 neighborhoods

$$CostMLFMM = (M+N)P + (P_4(d)+2)\frac{N}{s}CostTranslation(P) + P_2(d)sMCostFunc$$

Optimization of the Grouping Parameter



$$CostMLFMM = (M+N)P + (P_4(d)+2)\frac{N}{s}CostTranslation(P) + P_2(d)sMCostFunc$$

$$\frac{\partial CostMLFMM}{\partial s} = -(P_4(d)+2)\frac{N}{s^2}CostTranslation(P) + P_2(d)MCostFunc = 0$$

$$s_{opt} = \left[\frac{N(P_4(d)+2)CostTranslation(P)}{MP_2(d)CostFunc}\right]^{1/2}.$$

$$CostMLFMM_{opt} = (M+N)P + 2[MN(P_4(d)+2)P_2(d)CostTranslation(P)CostFunc]^{1/2}.$$

Optimization of the Grouping Parameter (Example)

$$s_{opt} = \left[\frac{N(P_4(d)+2)CostTranslation(P)}{MP_2(d)CostFunc}\right]^{1/2}.$$

$$CostMLFMM_{opt} = (M+N)P + 2[MN(P_4(d)+2)P_2(d)CostTranslation(P)CostFunc]^{1/2}.$$

Example:

$$N = M, \quad P_4(d) = 3^d\left(2^d - 1\right), \quad P_2(d) = 3^d,$$

$$CostTranslation(P) = P^2, \quad CostFunc = 1$$

$$s_{opt} \sim 2^{d/2}P; \quad CostMLFMM_{opt} \sim 2NP\left(1 + 3^d 2^{d/2}\right)$$

$$For\ d = 2, \quad P = 10, \quad s_{opt} \sim 38, \quad CostMLFMM_{opt} \sim 38NP = 380N.$$

If non-optimized,

$$s = 1; \quad CostMLFMM_{opt} \sim NP\left(2 + 3^d 2^d P\right)$$

$$For\ d = 2, \quad P = 10, \quad s = 1, \quad CostMLFMM_{opt} \sim 360NP = 3600N.$$

In this example optimization results in about 10 times savings!

DEMO

- Yang Wang (wpwy@umiacs.umd.edu), "Java Implementation and Simulation of the Fast Multipole Method for 2-D Coulombic Potential Problems," AMSC 698R course project report, 2003.
- http://brigade.umiacs.umd.edu/~wpwy/applet/FmmApplet.html
- Seems to work with Mozilla and Netscape …IE has problems

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Some Numerical Experiments with MLFMM

N.A. Gumerov, R. Duraiswami & E.A. Borovikov

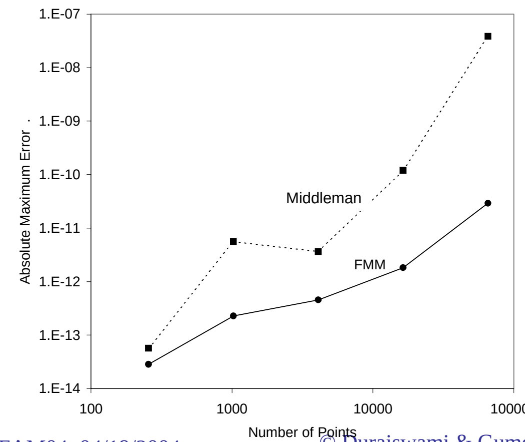
Data Structures, Optimal Choice of Parameters, and Complexity Results for Generalized Multilevel Fast Multipole Methods in *d* Dimensions.

UMIACS TR 2003-28,
Also issued as *Computer Science Technical Report CS-TR-# 4458*.
University of Maryland, College Park, 2003.

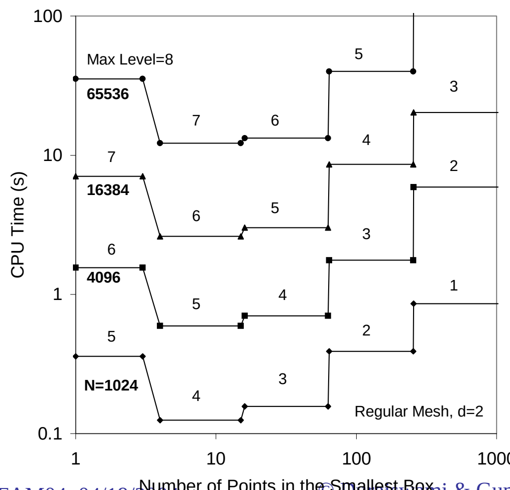
Available online via
http://www.umiacs.umd.edu/~ramani/pubs/umiacs-tr-2003-28.pdf

Error Test. FMM vs Middleman.

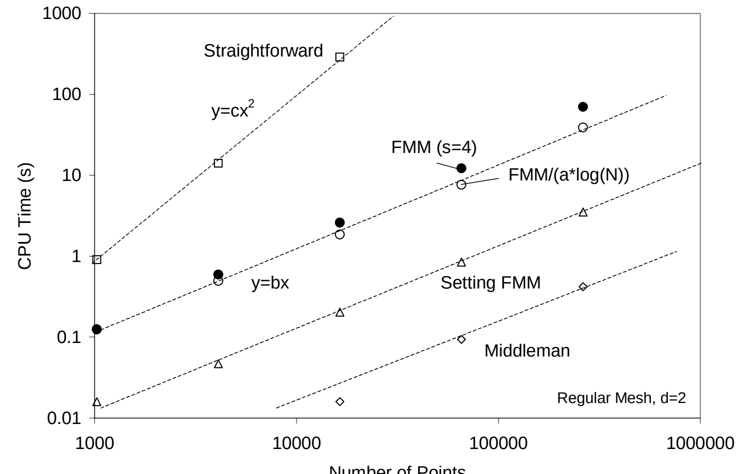
Regular Mesh, $N = M$.



Test with Varying Grouping Parameter.

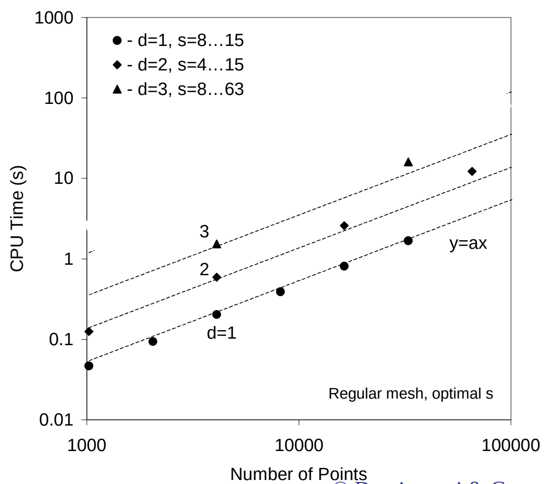


Test with Varying N.



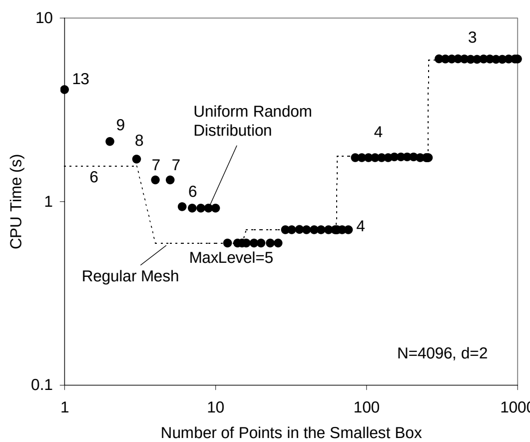
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Comparisons for different dimensionalities

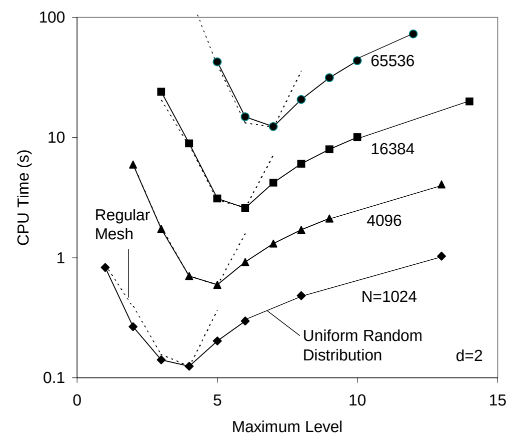


Random Distributions

Dependence of CPU Time on the Grouping Parameter, s



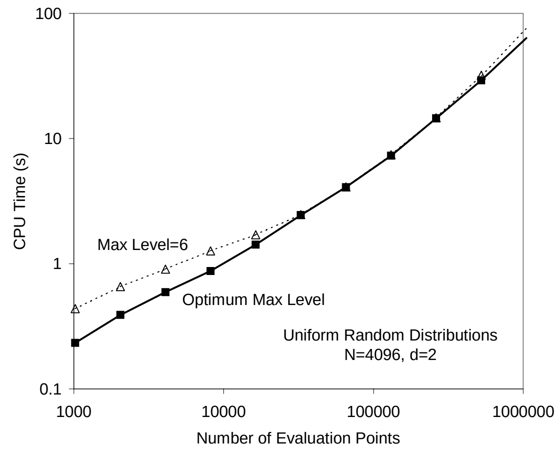
Dependence of CPU Time on the Maximum Space Subdivision Level



Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Dependence of CPU Time on *M*



Adaptive FMM

- H. Cheng, L. Greengard, and V. Rokhlin, "A Fast Adaptive Multipole Algorithms in Three Dimensions," Journal of Computational Physics, 155:468-498, 1999 .
- N.A. Gumerov, R. Duraiswami, and Y.A. Borovikov, "Data structures and algorithms for adaptive multilevel fast multipole methods," in preparation.

 © Duraiswami & Gumerov, 2003-2004

Fast Multipole Methods for The Laplace Equation

Ramani Duraiswami
Nail Gumerov

 © Duraiswami & Gumerov, 2003-2004

Outline

- 3D Laplace equation and Coulomb potentials
- Multipole and local expansions
- Special functions
  - Legendre polynomials
  - Associated Legendre functions
  - Spherical harmonics
- Translations of elementary solutions
- Complexity of FMM
- Reducing complexity
- Rotations of elementary solutions
- Coaxial Translation-Rotation decomposition
- Faster Translation techniques

 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Review

$$\mathbf{v} = \Phi \mathbf{u},$$

- FMM aims at accelerating the matrix vector product
- Matrix entries determined by a set of source points and evaluation points (possibly the same)

$$\Phi = \begin{pmatrix} \Phi(\mathbf{y}_1,\mathbf{x}_1) & \Phi(\mathbf{y}_1,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_1,\mathbf{x}_N) \\ \Phi(\mathbf{y}_2,\mathbf{x}_1) & \Phi(\mathbf{y}_2,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_2,\mathbf{x}_N) \\ \ldots & \ldots & \ldots & \ldots \\ \Phi(\mathbf{y}_M,\mathbf{x}_1) & \Phi(\mathbf{y}_M,\mathbf{x}_2) & \ldots & \Phi(\mathbf{y}_M,\mathbf{x}_N) \end{pmatrix}.$$

$$X = \{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\}, \quad \mathbf{x}_i \in R^d, \quad i = 1,\ldots,N,$$
$$Y = \{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_M\}, \quad \mathbf{y}_j \in R^d, \quad j = 1,\ldots,M.$$

- Function Φ has following point-centered representations about a given point x_*
  - ❑ Local (valid in a neighborhood of a given point)
  - ❑ Far-field or multipole (valid outside a neighborhood of a given point)
  - ❑ In many applications Φ is singular

$$v_j = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}_j,\mathbf{x}_i), \quad j = 1,\ldots,M.$$

- Representations are usually series
  - ❑ Could be integral transform representations
- Representations are usually approximate
  - ❑ Error bound guarantees the error is below a specified tolerance

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Review

$$\Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j) + Error(p,\mathbf{x}_i,\mathbf{y}_j).$$

- One representation, valid in a given domain, can be converted to another valid in a subdomain contained in the original domain
- Factorization trick is at core of the FMM speed up

$$v_j = \sum_{i=1}^{N} u_i \Phi(\mathbf{y}_j,\mathbf{x}_i) = \sum_{i=1}^{N} u_i \sum_{m=0}^{p-1} A_m(\mathbf{x}_i)F_m(\mathbf{y}_j) + \sum_{i=1}^{N} u_i Error(p,\mathbf{x}_i,\mathbf{y}_j)$$
$$= \sum_{m=0}^{p-1} B_m F_m(\mathbf{y}_j) + Error_j(p,N), \quad j = 1,\ldots,M.$$

- Representations we use are factored … separate points x_i and y_j
- Data is partitioned to organize the source points and evaluation points so that for each point we can separate the points over which we can use the factorization trick, and those we cannot.
- Hierarchical partitioning allows use of different factorizations for different groups of points
- Accomplished via MLFMM discussed yesterday
- Today concrete example for Laplace equation/Coulomb potentials

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Solution of Laplace's equation

- Green's function for Laplace's equation

$$\nabla^2 G(\mathbf{x},\mathbf{y}) = \delta(\mathbf{x}-\mathbf{y}) \qquad G(\mathbf{x},\mathbf{y}) = -\frac{1}{4\pi \mathbf{x}-\mathbf{y}}$$

- Green's formula

$$\begin{aligned} \phi(\mathbf{y}) &= \int_\Omega \phi(\mathbf{x})\delta(\mathbf{x}-\mathbf{y})d^3x = \int_\Omega \phi(\mathbf{x})\nabla^2 G(\mathbf{x},\mathbf{y})d^3x \\ &= -\int_\Omega \nabla\phi(\mathbf{x})\cdot\nabla G(\mathbf{x},\mathbf{y})d^3x + \int_{\partial\Omega}\phi(\mathbf{x})\mathbf{n}\cdot\nabla G(\mathbf{x},\mathbf{y})dS_x \\ &= \int_\Omega \nabla^2\phi(\mathbf{x})\nabla G(\mathbf{x},\mathbf{y})d^3x + \int_{\partial\Omega}[\phi(\mathbf{x})\mathbf{n}\cdot\nabla G(\mathbf{x},\mathbf{y}) - \mathbf{n}\cdot\nabla\phi(\mathbf{x})G(\mathbf{x},\mathbf{y})]\,dS_x \end{aligned}$$

- Goal solve Laplace's equation with given boundary conditions
  - ❑ E.g. $\nabla^2 \phi = 0$ in Ω $\qquad \partial\phi/\partial n = f$ on $\partial\Omega$

$$\phi(\mathbf{y}) - \int_{\partial\Omega}\phi(\mathbf{x})\frac{\partial G}{\partial n}(\mathbf{x},\mathbf{y}) = -\int_{\partial\Omega} f(\mathbf{x})G(\mathbf{x},\mathbf{y})dS_x$$

- Upon discretization yields system of type that can be solved iteratively, with matrix vector products accelerated by FMM

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Molecular and stellar dynamics

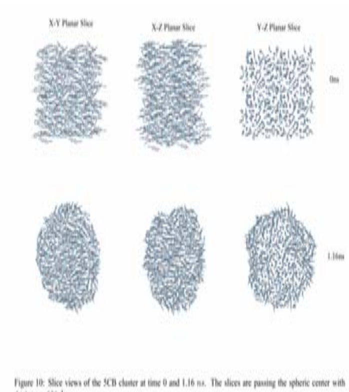
- Many particles distributed in space
- Particles are moving and exert a force on each other
- Simplest case this force obeys an inverse-square law (gravity, coulombic interaction)
- Goal of computations compute the dynamics

$$\frac{d^2\mathbf{x}_i}{dt^2} = F_i,$$

- Force is

$$F_i = \sum_{\substack{j=1 \\ j\neq i}}^{N} q_i q_j \frac{(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|^3}$$

- After time step, particles move
- Recompute force and iterate



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

What is needed for the FMM

- Local expansion
- Far-field or multipole expansion
- Translations
  - Multipole-to-multipole (S|S)
  - Local-to-local (R|R)
  - Multipole-to-local (S|R)
- Error bounds

Translation and Differentiation Properties for Laplace Equation

If

$$\nabla^2 \Phi(\mathbf{r}) = 0, \quad \mathbf{r} \in \Omega.$$

then shifted function $\Phi(\mathbf{r} - \mathbf{r}_0)$ also satisfies the Laplace equation

$$\nabla^2 \Phi(\mathbf{r} - \mathbf{r}_0) = 0, \quad \mathbf{r} - \mathbf{r}_0 \in \Omega.$$

Also the Laplace operator is commutative with differential operators

$$D_x = \frac{\partial}{\partial x}, \quad D_y = \frac{\partial}{\partial y}, \quad D_z = \frac{\partial}{\partial z}, \quad \text{or} \quad D_{\mathbf{t}} = \mathbf{t} \cdot \nabla,$$

So

$$D_{\mathbf{t}} \nabla^2 \Phi(\mathbf{r}) = \nabla^2 D_{\mathbf{t}} \Phi(\mathbf{r}).$$

Introduction of Multipoles for Laplace Equation

$$\Phi_n(\mathbf{r}) = (-1)^n D_{\mathbf{t}_1} D_{\mathbf{t}_2} \ldots D_{\mathbf{t}_n} \Phi(\mathbf{r})$$

also satisfy the Laplace equation. In case when $\Phi(\mathbf{r}) = G(\mathbf{r}) = |\mathbf{r}|^{-1}$ functions

$$G_n(\mathbf{r}) = (-1)^n D_{\mathbf{t}_1} D_{\mathbf{t}_2} \ldots D_{\mathbf{t}_n} \frac{1}{|\mathbf{r}|}, \quad |\mathbf{r}| = \sqrt{x^2 + y^2 + z^2} \neq 0$$

are called MULTIPOLES OF DEGREE n centered at $\mathbf{r} = \mathbf{0}$. Vectors $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ are called multole generating vectors. Also $G_n(\mathbf{r})$ can be represented as

$$G_n(\mathbf{r}) = \sum_{i+j+k=n} Q_{ijk}^{(n)} \frac{\partial^n}{\partial x^i \partial y^j \partial z^k} \frac{1}{|\mathbf{r}|},$$

where $Q_{ijk}^{(n)}$ are called 'components of the multipole momentum'.

$n = 0$: 'monopole'
$n = 1$: 'dipole'
$n = 2$: 'quadrupole'
$n = 3$: 'octupole'.

Multipole Expansion of Laplace Equation Solutions

$$\Phi(\mathbf{r}) = \sum_{n=0}^{\infty} b_n G_n(\mathbf{r}),$$

$$G_n(\mathbf{r}) = \sum_{i+j+k=n} Q_{ijk}^{(n)} \frac{\partial^n}{\partial x^i \partial y^j \partial z^k} \frac{1}{|\mathbf{r}|}.$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Legendre Polynomials

Legendre polynomials $P_n(\mu)$ can be introduced via generating function

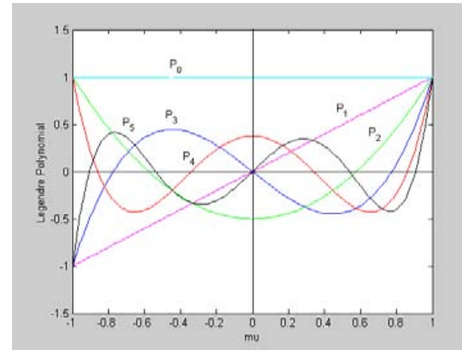
$$\frac{1}{\sqrt{1-2\mu x+x^2}} = \begin{cases} \sum_{n=0}^{\infty} P_n(\mu) x^n, & |x| < 1, \\ \sum_{n=0}^{\infty} P_n(\mu) x^{-n-1}, & |x| > 1. \end{cases}$$

First few polynomials

$$P_0(\mu) = 1,$$
$$P_1(\mu) = \mu = \cos\theta,$$
$$P_2(\mu) = \frac{1}{2}(3\mu^2 - 1) = \frac{1}{4}(3\cos 2\theta + 1),$$
$$\ldots$$

Legendre Polynomials (2)

First six polynomials ($n = 0,\ldots,5$):



Legendre Polynomials (3)

Some Properties:

- The Rodrigues' formula

$$P_n(\mu) = \frac{1}{2^n n!} \frac{d^n}{d\mu^n} (\mu^2 - 1)^n.$$

- Form orthogonal complete basis in $L_2[-1,1]$:

$$\int_{-1}^{1} P_n(\mu) P_m(\mu) d\mu = \begin{cases} \frac{2}{2n+1}, & m = n, \\ 0, & m \neq n. \end{cases}$$

A lot of other nice properties!

Expansion/Translation of Fundamental Solution

$$G(\mathbf{r}) = \frac{1}{r}, \quad r = |\mathbf{r}| = \sqrt{x^2+y^2+z^2},$$

then

$$G(\mathbf{r}-\mathbf{r}_0) = \frac{1}{|\mathbf{r}-\mathbf{r}_0|} = \frac{1}{\sqrt{(\mathbf{r}-\mathbf{r}_0)\cdot(\mathbf{r}-\mathbf{r}_0)}} = \frac{1}{\sqrt{r^2 - 2\mathbf{r}\cdot\mathbf{r}_0 + r_0^2}}$$
$$= \frac{1}{\sqrt{r^2 - 2rr_0\cos\theta + r_0^2}} = \frac{1}{\sqrt{r^2 - 2\mu r r_0 + r_0^2}}$$
$$= \begin{cases} r_0^{-1} \sum_{n=0}^{\infty} P_n(\mu)(r/r_0)^n, & r < r_0, \\ r^{-1} \sum_{n=0}^{\infty} P_n(\mu)(r_0/r)^n = r_0^{-1} \sum_{n=0}^{\infty} P_n(\mu)(r/r_0)^{-n-1}, & r > r_0. \end{cases}$$

At $r = r_0$ the series also converges, if $\cos\theta \neq 1$ ($\mathbf{r} \neq \mathbf{r}_0$).

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Addition Theorem for Spherical Harmonics

Spherical Harmonics

$$P_n(\cos\theta) = \frac{4\pi}{2n+1}\sum_{m=-n}^{n} Y_n^{-m}(\theta',\varphi')Y_n^m(\hat\theta,\hat\varphi),$$

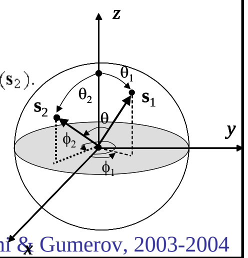
order

$$Y_n^m(\theta,\varphi) = (-1)^m\sqrt{\frac{2n+1}{4\pi}\frac{(n-|m|)!}{(n+|m|)!}}P_n^{|m|}(\mu)e^{im\varphi}, \quad \mu = \cos\theta.$$

where θ is the angle between two points on a sphere with spherical angles (θ',φ') and $(\hat\theta,\hat\varphi)$.

degree

$$P_n(\mathbf{s}_1\cdot\mathbf{s}_2) = \frac{4\pi}{2n+1}\sum_{m=-n}^{n} Y_n^{-m}(\mathbf{s}_1)Y_n^m(\mathbf{s}_2) = \frac{4\pi}{2n+1}\sum_{m=-n}^{n} Y_n^m(\mathbf{s}_1)Y_n^{-m}(\mathbf{s}_2).$$



Vector form of the addition theorem

Associated Legendre Functions

$$P_n^m(\mu) = \frac{(-1)^m}{2^m}\frac{(n+m)!}{(n-m)!m!}(1-\mu^2)^{m/2}F\left(m-n, m+n+1; m+1; \frac{1-\mu}{2}\right)$$
$$= \frac{(-1)^m}{2^m}\frac{(n+m)!}{(n-m)!m!}(1-\mu^2)^{m/2}\sum_{l=0}^{n-m}\frac{(-1)^l(n-m-l+1)_l(n+m+1)_l}{2^l l!(m+1)_l}(1-\mu)^l,$$

where $(n)_l$ is the Pochhammer's symbol:

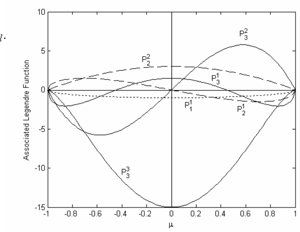
$$(n)_0 = 1, \quad (n)_l = \frac{(n+l-1)!}{(n-1)!}.$$

This formula yields the following particular functions:

$$P_1^1(\mu) = -(1-\mu^2)^{1/2}, \quad P_2^1(\mu) = -3\mu(1-\mu^2)^{1/2}, \quad P_2^2(\mu) = 3(1-\mu^2).$$

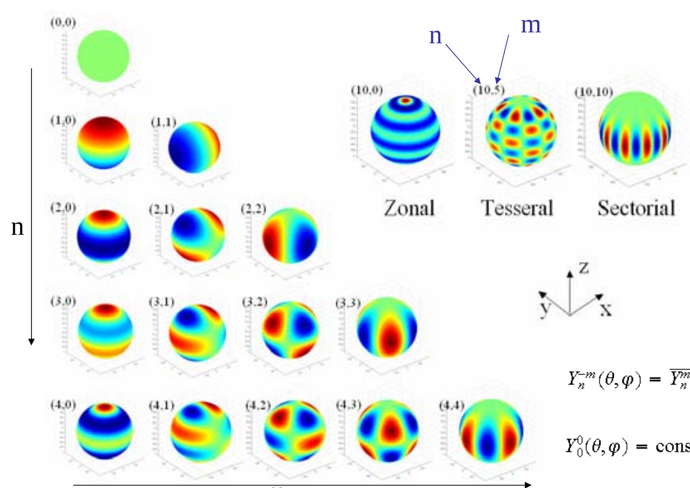
$$(P_n^m, P_l^m) = \int_{-1}^{1}P_n^m(\mu)P_l^m(\mu)d\mu = \frac{2}{2n+1}\frac{(n+m)!}{(n-m)!}\delta_{nl}.$$

Orthogonal!



Spherical Harmonics

$$Y_n^m(\theta,\varphi) = (-1)^m\sqrt{\frac{2n+1}{4\pi}\frac{(n-|m|)!}{(n+|m|)!}}P_n^{|m|}(\cos\theta)e^{im\varphi},$$
$$n = 0,1,2,\ldots; \quad m = -n,\ldots,n.$$



$$Y_n^{-m}(\theta,\varphi) = \overline{Y_n^m(\theta,\varphi)}.$$

$$Y_0^0(\theta,\varphi) = \text{const} = \sqrt{\frac{1}{4\pi}}.$$

Orthonormality of Spherical Harmonics

The scalar product of two spherical harmonics in $L_2(S_u)$ is

$$\left(Y_n^m, Y_{n'}^{m'}\right) = \int_0^\pi \sin\theta d\theta\int_0^{2\pi} Y_n^m(\theta,\varphi)\bar{Y}_{n'}^{m'}(\theta,\varphi)d\varphi = \delta_{mm'}\delta_{nn'}.$$

Expansion of an arbitrary surface function over the basis of spherical harmonics:

$$F(\theta,\varphi) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}F_n^m Y_n^m(\theta,\varphi).$$

$$\left(F, Y_{n'}^{m'}\right) = \int_0^\pi \sin\theta d\theta\int_0^{2\pi} F(\theta,\varphi)Y_{n'}^{-m'}(\theta,\varphi)d\varphi.$$

$$\left(F, Y_{n'}^{m'}\right) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}F_n^m\left(Y_n^m, Y_{n'}^{m'}\right) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}F_n^m\delta_{mm'}\delta_{nn'} = F_{n'}^{m'}.$$

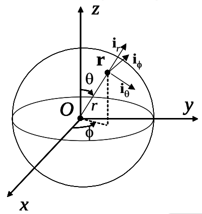
$$F_n^{m'} = \int_0^\pi \sin\theta d\theta\int_0^{2\pi} F(\theta,\varphi)Y_{n'}^{-m'}(\theta,\varphi)d\varphi.$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

S- and R- expansions of Fundamental Solution

$$\frac{1}{|\mathbf{r}-\mathbf{r}_0|} = \frac{4\pi}{r_0}\sum_{n=0}^{\infty}\frac{1}{2n+1}\left(\frac{r}{r_0}\right)^n \sum_{m=-n}^{n} Y_n^{-m}(\theta',\varphi')Y_n^m\left(\hat{\theta},\hat{\varphi}\right), \quad r < r_0,$$

$$\frac{1}{|\mathbf{r}-\mathbf{r}_0|} = \frac{4\pi}{r_0}\sum_{n=0}^{\infty}\frac{1}{2n+1}\left(\frac{r_0}{r}\right)^{n+1} \sum_{m=-n}^{n} Y_n^{-m}(\theta',\varphi')Y_n^m\left(\hat{\theta},\hat{\varphi}\right), \quad r > r_0.$$



$$\mathbf{r} = r(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta),$$

$$R_n^m(\mathbf{r}) = r^n Y_n^m(\theta,\varphi),$$
$$S_n^m(\mathbf{r}) = r^{-n-1} Y_n^m(\theta,\varphi),$$

Multipole (!)

$$\frac{1}{4\pi|\mathbf{r}-\mathbf{r}_0|} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{1}{2n+1}S_n^{-m}(\mathbf{r}_0)R_n^m(\mathbf{r}), \quad r < r_0,$$

$$\frac{1}{4\pi|\mathbf{r}-\mathbf{r}_0|} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{1}{2n+1}R_n^{-m}(\mathbf{r}_0)S_n^m(\mathbf{r}), \quad r > r_0.$$

Error bound

- Series converge rapidly
  - ❑E.g,. For multipole expansion we have

$$\Phi(P) = \sum_{i=1}^{k}\frac{q_i}{\|P_i - P\|}$$

potential due to a set of k sources of strengths $\{q_i,\ i = 1,\ldots,k\}$ at $\{P_i = (r_i,\theta_i,\phi_i),\ i = 1,\ldots,k\}$, with $|r_i| < a$. Then for $P = (r,\theta,\phi) \in R^3$ with $|r| > a$,

$$\Phi(P) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}}Y_n^m(\theta,\phi),$$

$$M_n^m = \sum_{i=1}^{k}(-1)^m q_i * r_i^n * Y_n^{-m}(\theta_i,\phi_i).$$

$$\left|\Phi(P) - \sum_{n=0}^{p}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}}Y_n^m(\theta,\phi)\right| \le \frac{A}{r-a}\left(\frac{a}{r}\right)^{p+1},$$

$$A = \sum_{i=1}^{k}|q_i|.$$

`Multipole expansion` is S-expansion

Compare

$$\frac{1}{4\pi|\mathbf{r}-\mathbf{r}_0|} = \sum_{n=0}^{\infty}b_n G_n(\mathbf{r}), \quad G_n(\mathbf{r}) = \sum_{i+j+k=n}Q_{ijk}^{(n)}\frac{\partial^n}{\partial x^i\partial y^j\partial z^k}\frac{1}{|\mathbf{r}|}.$$

and

$$\frac{1}{4\pi|\mathbf{r}-\mathbf{r}_0|} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{1}{2n+1}R_n^{-m}(\mathbf{r}_0)S_n^m(\mathbf{r}), \quad r > r_0.$$

$$b_n\sum_{i+j+k=n}Q_{ijk}^{(n)}\frac{\partial^n}{\partial x^i\partial y^j\partial z^k}\frac{1}{|\mathbf{r}|} = \sum_{m=-n}^{n}\frac{1}{2n+1}R_n^{-m}(\mathbf{r}_0)S_n^m(\mathbf{r}).$$

Generally

$$\sum_{i+j+k=n}Q_{ijk}^{(n)}\frac{\partial^n}{\partial x^i\partial y^j\partial z^k}\frac{1}{|\mathbf{r}|} = \sum_{m=-n}^{n}q_n^m S_n^m(\mathbf{r}) = \frac{1}{r^{n+1}}\sum_{m=-n}^{n}q_n^m Y_n^m(\theta,\varphi).$$

R- and S- expansions of arbitrary solutions of the 3D Laplace equation

$$\Phi(\mathbf{r}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left[A_n^m R_n^m(\mathbf{r}) + B_n^m S_n^m(\mathbf{r})\right],$$

Functions regular at $\mathbf{r} = \mathbf{0}$:

$$\Phi(\mathbf{r}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}A_n^m R_n^m(\mathbf{r}),$$

Functions decaying at $|\mathbf{r}| \to \infty$:

$$\Phi(\mathbf{r}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}B_n^m S_n^m(\mathbf{r}).$$

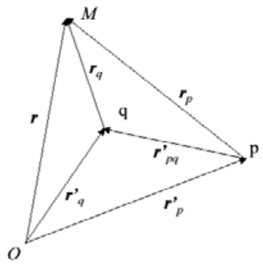
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Translations of elementary solutions of the 3D Laplace equation

$$S_n^m(\mathbf{r}_p) = \sum_{l=0}^{\infty}\sum_{s=-l}^{l}(S|R)_{ln}^{sm}(\mathbf{r}'_{pq})R_l^s(\mathbf{r}_q), \qquad |\mathbf{r}_q| < |\mathbf{r}'_{pq}|, \quad p \neq q.$$

$$S_n^m(\mathbf{r}_p) = \sum_{l=0}^{\infty}\sum_{s=-l}^{l}(S|S)_{ln}^{sm}(\mathbf{r}'_{pq})S_l^s(\mathbf{r}_q), \qquad |\mathbf{r}_q| > |\mathbf{r}'_{pq}|,$$

$$R_n^m(\mathbf{r}_p) = \sum_{l=0}^{\infty}\sum_{s=-l}^{l}(R|R)_{ln}^{sm}(\mathbf{r}'_{pq})R_l^s(\mathbf{r}_q).$$



For a p-truncated expansion *(E|F)* is a $p^2 \times p^2$ matrix

See Tang 03 or Greengard 89 for explicit expressions

Translation of a Multipole Expansion

Let

$$\Phi(P) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{O_n^m}{r'^{n+1}}Y_n^m(\theta',\phi'),$$

Where $P - Q = (r', \theta', \phi')$. Then the potential ϕ can be expressed as,

$$\Phi(P) = \sum_{j=0}^{\infty}\sum_{k=-j}^{j}\frac{M_j^k}{r^{j+1}}Y_j^k(\theta,\phi),$$

$$M_j^k = \sum_{n=0}^{j}\sum_{m=\max(k+n-j,-n)}^{\min(k+j-n,n)}\frac{O_{j-n}^{k-m} i^{|k|-|m|-|k-m|}A_n^m A_{j-n}^{k-m}\rho^n Y_n^{-m}(\alpha,\beta)}{A_j^k},$$

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}. \qquad M = SS(\rho,\alpha,\beta) * O$$

Translation of a Local Expansion

Suppose that

$$\Phi(P) = \sum_{n=0}^{p}\sum_{m=-n}^{n}O_n^m r'^n Y_n^m(\theta',\phi')$$

is a local expansion centered at $Q = (\rho, \alpha, \beta)$, Where $P = (r, \theta, \phi)$, and $P - Q = (r', \theta', \phi')$. Then the local expansion centered at origin is

$$\Phi(P) = \sum_{j=0}^{p}\sum_{k=-j}^{j}L_j^k r^j Y_j^k(\theta,\phi),$$

where

$$L_j^k = \sum_{n=j}^{p}\sum_{m=k-n+j}^{k-j+n}\frac{O_n^m i^{|m|-|m-k|-|k|}A_j^k A_{n-j}^{m-k}\rho^{n-j}Y_{n-j}^{m-k}(\alpha,\beta)}{(-1)^{n+j}A_n^m},$$

$$L = RR(\rho,\alpha,\beta) * O$$

Complexity Analysis

Step 1,Forming Expansions $O(Np^2)$.
Step 2, Upward pass with Matrix based S|S translations

$$\sum_{l=2}^{n-1} 8 * 8^l * p^4 = \frac{8^3 - 8^{n+1}}{1-8}p^4 \approx \frac{8}{7}8^n p^4 = \frac{8}{7}\frac{N}{s}p^4.$$

Step 3, Downward pass with matrix based S|R and R|R translations

$$\sum_{l=2}^{n} 8^l * p^4 + \sum_{l=2}^{n} 8^l * p^4 * 189 \approx \frac{8}{7} * 8^n * 190p^4 = \frac{1520}{7}\frac{N}{s}p^4.$$

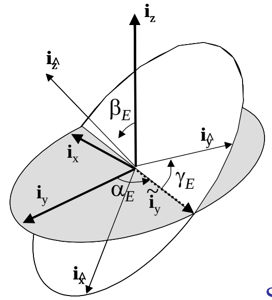
Step 4, Evaluate R expansions at points$O(Np^2)$
Step 5, Sum missed neighbor points $O(27Ns)$
The total cost for all five steps is approximately

$$2Np^2 + \frac{1528}{7}\frac{N}{s}p^4 + 27Ns.$$

With $s \approx \sqrt{\frac{1528}{189}}p^2$, the total number of operations is approximately $156Np^2$.

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Rotations of coordinates



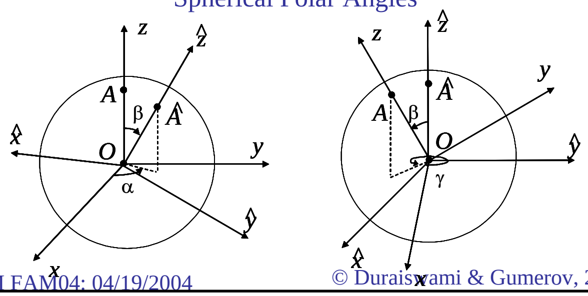
Rotation Matrix

$$Q = \begin{bmatrix} \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_z \\ \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_z \\ \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_z \end{bmatrix}$$

Euler Angles

$$\alpha_E = \pi - \alpha, \quad \beta_E = \beta, \quad \gamma_E = \gamma.$$

Spherical Polar Angles



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Rotations of elementary solutions of the 3D Laplace equation

Rotations

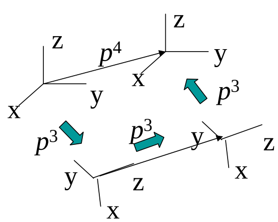
$$Y_n^m(\theta,\varphi) = \sum_{\nu=-n}^{n} T_n^{\nu m}(Q) Y_n^{\nu}\left(\hat{\theta},\hat{\varphi}\right),$$

$$S_n^m(\mathbf{r}_p) = \sum_{\nu=-n}^{n} T_n^{\nu m}(Q) S_n^{\nu}(\hat{\mathbf{r}}_p), \qquad |\hat{\mathbf{r}}_p| = |\mathbf{r}_p|,$$

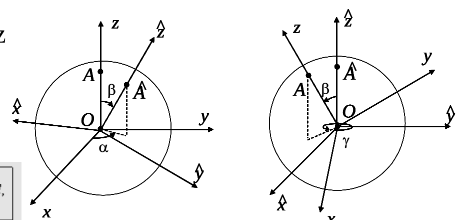
$$R_n^m(\mathbf{r}_p) = \sum_{\nu=-n}^{n} T_n^{\nu m}(Q) R_n^{\nu}(\hat{\mathbf{r}}_p), \qquad |\hat{\mathbf{r}}_p| = |\mathbf{r}_p|,$$

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Rotation-Coaxial Translation Decomposition



Rotation



Coaxial Translation

$$S_n^m(\tilde{\mathbf{r}} + \mathbf{i}_z d) = \sum_{l=|m|}^{\infty} (S|R)_{ln}^m(d) R_l^m(\tilde{\mathbf{r}}), \qquad |\tilde{\mathbf{r}}| < d,$$

$$S_n^m(\tilde{\mathbf{r}} + \mathbf{i}_z d) = \sum_{l=|m|}^{\infty} (S|S)_{ln}^m(d) S_l^m(\tilde{\mathbf{r}}), \qquad |\tilde{\mathbf{r}}| > d,$$

$$R_n^m(\tilde{\mathbf{r}} + \mathbf{i}_z d) = \sum_{l=|m|}^{\infty} (R|R)_{ln}^m(d) R_l^m(\tilde{\mathbf{r}}).$$

$$(E|F)_{ln}^m(d) = (E|F)_{ln}^{mm}(\mathbf{d})|_{\theta'_{pq}=0}, \quad E,F = S,R.$$

$$Y_n^m(\theta,\varphi) = \sum_{\nu=-n}^{n} T_n^{\nu m}(Q) Y_n^{\nu}\left(\hat{\theta},\hat{\varphi}\right),$$

$$Q = \begin{bmatrix} \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_z \\ \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_z \\ \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_z \end{bmatrix}$$

© Duraiswami & Gumerov, 2003-2004

Coaxial translation operator has invariant subspaces at fixed order, *m*, while the rotation operator has invariant subspaces at fixed degree, *n*.

Coaxial Translation:

$$(\mathbf{S}|\mathbf{R}) = (\mathbf{S}|\mathbf{R})^0 \oplus (\mathbf{S}|\mathbf{R})^{\pm 1} \oplus \ldots = \sum_{m=-\infty}^{\infty} \oplus (\mathbf{S}|\mathbf{R})^m,$$

Rotation

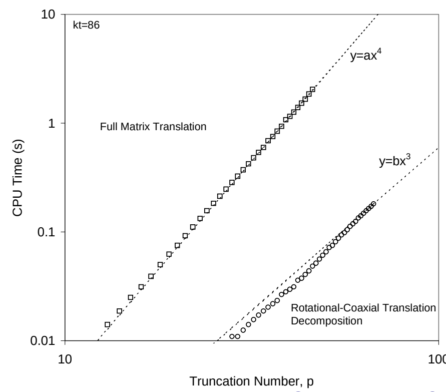
$$(\mathbf{S}|\mathbf{R}) = (\mathbf{S}|\mathbf{R})_0 \oplus (\mathbf{S}|\mathbf{R})_1 \oplus \ldots = \sum_{n=0}^{\infty} \oplus (\mathbf{S}|\mathbf{R})_n,$$

Each can be done in p operations which cost $O(p^2)$ resulting in $O(p^3)$ complexity

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Comparison of Direct Matrix Translation and Coaxial Translation-Rotation Decomposition



Other Fast translation schemes: Elliot and Board (1996)

Renormalized S- and R- functions

Definition:

$$\widetilde{S}_n^m(\mathbf{r}) = O_n^m(\mathbf{r}) = \frac{(-1)^n i^{|m|}}{\alpha_n^m}\sqrt{\frac{4\pi}{2n+1}}\, S_n^m(\mathbf{r}) = \frac{(-1)^n i^{|m|}}{\alpha_n^m}\sqrt{\frac{4\pi}{2n+1}}\,\frac{1}{r^{n+1}}\, Y_n^m(\theta,\varphi),$$

$$\widetilde{R}_n^m(\mathbf{r}) = I_n^m(\mathbf{r}) = i^{-|m|}\alpha_n^m\sqrt{\frac{4\pi}{2n+1}}\, R_n^m(\mathbf{r}) = i^{-|m|}\alpha_n^m\sqrt{\frac{4\pi}{2n+1}}\, r^n Y_n^m(\theta,\varphi),$$

where

$$\alpha_n^m = \alpha_n^{-m} = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}.$$

Other Fast translation schemes: Elliot and Board (1996)

In the renormalized basis translation matrices are simple

$$(\widetilde{S}|\widetilde{R})_{n'n}^{m'm}(\mathbf{t}) = (O|I)_{n'n}^{m'm}(\mathbf{t}) = O_{n+n'}^{m-m'}(\mathbf{t}) = \widetilde{S}_{n+n'}^{m-m'}(\mathbf{t}),$$

$$(\widetilde{S}|\widetilde{S})_{n'n}^{m'm}(\mathbf{t}) = (O|O)_{n'n}^{m'm}(\mathbf{t}) = I_{n'-n}^{m-m'}(\mathbf{t}) = \widetilde{R}_{n'-n}^{m-m'}(\mathbf{t}),$$

$$(\widetilde{R}|\widetilde{R})_{n'n}^{m'm}(\mathbf{t}) = (I|I)_{n'n}^{m'm}(\mathbf{t}) = I_{n-n'}^{m-m'}(\mathbf{t}) = \widetilde{R}_{n-n'}^{m-m'}(\mathbf{t}).$$

These are structured matrices (2D Toeplitz-Hankel type)
Fast translation procedures are possible
(e.g. see $O(p^2 \log p)$ algorithm in **W.D. Elliott & J.A. Board, Jr.:** ``Fast Fourier Transform Accelerated Fast Multipole Algorithm" *SIAM J. Sci. Comput.* Vol. 17, No. 2, pp. 398-415, 1996).
However, there are some stability issues reported.

Structured matrix based translation

- Tang 03
- Idea: use the rotation-coaxial translation method, and decompose resulting matrices into structured matrices
- Cost $O(p^2 \log p)$
- Details in Tang's thesis.

http://www.umiacs.umd.edu/~ramani/pubs/zhihui_thesis.pdf

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Complexity

The total cost of the original algorithm is

$$2Np^2 + \frac{1528}{7}\frac{N}{s}p^4 + 27Ns.$$

With $s \approx \sqrt{\frac{1528}{189}}p^2$, it is $156Np^2$.
In Tang's algorithm, the total cost is

$$2Np^2 + \frac{1528}{7}\frac{N}{s} * \frac{85}{4}p^2\log(4p) + 9Ns.$$

With $s \approx \frac{\sqrt{228480p^2\log(4p)}}{21}$, it is

$$2Np^2 + 410\sqrt{\log(4p)}Np.$$

According to this result, the break even p is 5.

Cheng et al 1999

- H. Cheng,¤ L. Greengard,*y* and V. Rokhlin, A Fast Adaptive Multipole Algorithm in Three Dimensions, Journal of Computational Physics **155,** 468–498 (1999)
- Convert to a transform representation ("plane-wave")
  - at a cost of O(p^2 log p)
  - Expansion formula

$$\frac{1}{r} = \frac{1}{2\pi}\int_0^\infty e^{-\lambda(z-z_0)}\int_0^{2\pi} e^{i\lambda((x-x_0)\cos\alpha+(y-y_0)\sin\alpha)}\,d\alpha\,d\lambda.$$

- Discretize integrals

$$\left|\frac{1}{r} - \sum_{k=1}^{s(\varepsilon)}\frac{w_k}{M_k}\sum_{j=1}^{M_k} e^{-\lambda_k\cdot(z-z_0)}\cdot e^{i\lambda_k\cdot[(x-x_0)\cdot\cos(\alpha_{j,k})+(y-y_0)\cdot\sin(\alpha_{j,k})]}\right| < \varepsilon,$$

- Trans
- Convert back

The FMM for 3D Helmholtz Equation

Nail Gumerov &
Ramani Duraiswami
UMIACS
[gumerov][ramani]@umiacs.umd.edu

Reference

N.A. Gumerov & R. Duraiswami

Fast Multipole Methods for Solution of the Helmholtz Equation in Three Dimensions

Academic Press, Oxford (2004)
(in process).

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Content

Helmholtz Equation

Helmholtz Equation

$$\nabla^2 \psi + k^2 \psi = 0$$

- Wave equation in frequency domain
  - Acoustics
  - Electromagneics (Maxwell equations)
  - Diffusion/heat transfer/boundary layers
  - Telegraph, and related equations
  - *k* can be complex
- Quantum mechanics
  - Klein-Gordan equation
  - Shroedinger equation
- Relativistic gravity (Yukawa potentials, *k* is purely imaginary)
- Molecular dynamics (Yukawa)
- Appears in many other models

Boundary Value Problems

- Dirichlet:

$$\psi|_S = 0,$$

- Neumann:

$$\left.\frac{\partial \psi}{\partial n}\right|_S = 0,$$

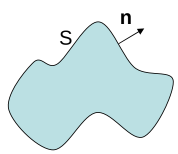
- Robin:

$$\left.\left(\frac{\partial \psi}{\partial n} + i\sigma\psi\right)\right|_S = 0.$$

- Sommerfield Radiation Condition (for external problems):

$$\psi = \psi_{in} + \psi_{scat}$$

$$\lim_{r\to\infty}\left[r\left(\frac{\partial \psi_{scat}}{\partial r} - ik\psi_{scat}\right)\right] = 0.$$



gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Green's Function and Identity

Free space Green's function:

$$\nabla^2 G(\mathbf{x},\mathbf{y}) + k^2 G(\mathbf{x},\mathbf{y}) = -\delta(\mathbf{x}-\mathbf{y}),$$

$$G(\mathbf{x},\mathbf{y}) = \frac{\exp(ik|\mathbf{x}-\mathbf{y}|)}{4\pi|\mathbf{x}-\mathbf{y}|}, \quad \mathbf{x},\mathbf{y} \in \mathbb{R}^3.$$

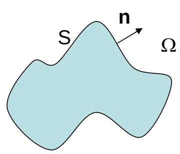
Green's formula:

$$\psi(\mathbf{y}) = \int_S \left[\psi(\mathbf{x})\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{x})} - G(\mathbf{x},\mathbf{y})\frac{\partial \psi(\mathbf{x})}{\partial n(\mathbf{x})}\right] dS(\mathbf{x}), \quad \mathbf{y} \in \Omega.$$

Boundary integral equation

$$\alpha\psi(\mathbf{y}) = \int_S \left(\psi(\mathbf{x})\frac{\partial G(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{x})} - G(\mathbf{x},\mathbf{y})\frac{\partial \psi(\mathbf{x})}{\partial n(\mathbf{x})}\right) dS(\mathbf{x}),$$

$$\alpha = \begin{cases} \frac{1}{2} & \mathbf{y} \text{ on a smooth part of the boundary} \\ \frac{\gamma}{4\pi} & \mathbf{y} \text{ at a corner on the boundary} \\ 1 & \mathbf{y} \text{ inside the domain} \end{cases}$$



Distributions of Monopoles and Dipoles

Volume source distribution:

$$\psi(\mathbf{y}) = \sum_{j=1}^{N} Q_j G(\mathbf{x}_j,\mathbf{y}), \quad \mathbf{y} \in \mathbb{R}^3 \backslash \{\mathbf{x}_j\},$$

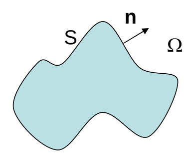
$$\psi(\mathbf{y}) = \int_{\Omega'} q(\mathbf{x}) G(\mathbf{x},\mathbf{y}) dV(\mathbf{x}), \quad \mathbf{y} \in \Omega, \quad \overline{\Omega'} \cap \Omega = \emptyset.$$

Single layer potential:

$$\psi(\mathbf{y}) = \int_S q_\sigma(\mathbf{x}) G(\mathbf{x},\mathbf{y}) dS(\mathbf{x}), \quad \mathbf{y} \in \Omega, \quad S = \partial\Omega.$$

Double layer potential:

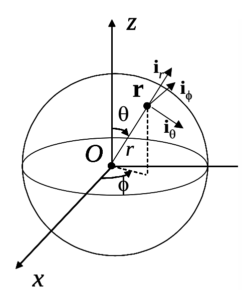
$$\psi(\mathbf{y}) = \int_S q_\mu(\mathbf{x}) \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{x})} dS(\mathbf{x}), \quad \mathbf{y} \in \Omega, \quad S = \partial\Omega.$$



Expansions in Spherical Coordinates

Spherical Basis Functions

$x = r\sin\theta\cos\varphi, \quad y = r\sin\theta\sin\varphi, \quad z = r\cos\theta.$



Spherical Coordinates

Spherical Bessel Functions

Regular Basis Functions

$$R_n^m(\mathbf{r}) = j_n(kr) Y_n^m(\theta,\varphi),$$

Singular Basis Functions

$$S_n^m(\mathbf{r}) = h_n(kr) Y_n^m(\theta,\varphi).$$

Spherical Hankel Functions of the First Kind

Spherical Harmonics

$$Y_n^m(\theta,\varphi) = (-1)^m \sqrt{\frac{2n+1}{4\pi}\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(\cos\theta) e^{im\varphi},$$

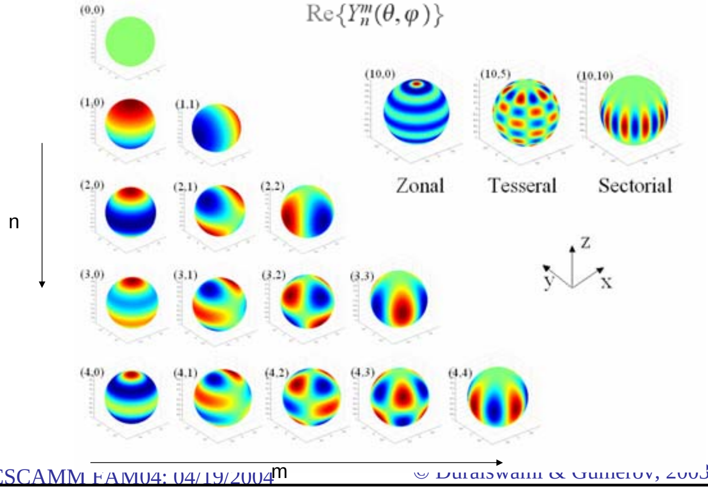
$$n = 0,1,2,\ldots; \qquad m = -n,\ldots,n.$$

Associated Legendre Functions

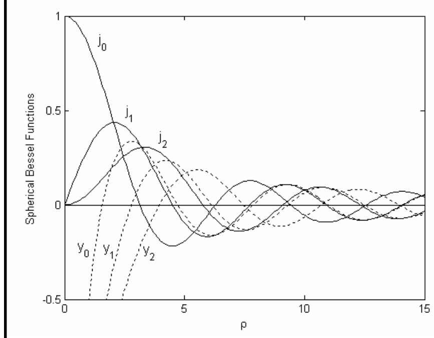
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Spherical Harmonics

$\mathrm{Re}\{Y_n^m(\theta,\varphi)\}$



Spherical Bessel Functions



$$h_n(\rho) = j_n(\rho) + iy_n(\rho)$$

$$j_0(\rho) = \frac{\sin\rho}{\rho}, \quad j_1(\rho) = \frac{\sin\rho}{\rho^2} - \frac{\cos\rho}{\rho},$$

$$j_2(\rho) = \left(\frac{3}{\rho^3} - \frac{1}{\rho}\right)\sin\rho - \frac{3}{\rho^2}\cos\rho,$$

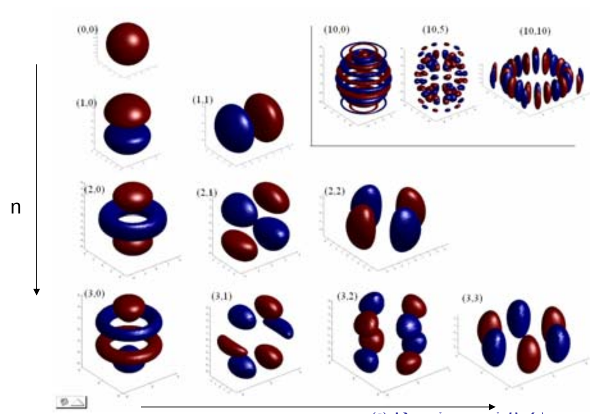
$$y_0(\rho) = -\frac{\cos\rho}{\rho}, \quad y_1(\rho) = -\frac{\cos\rho}{\rho^2} - \frac{\sin\rho}{\rho},$$

$$y_2(\rho) = \left(-\frac{3}{\rho^3} + \frac{1}{\rho}\right)\cos\rho - \frac{3}{\rho^2}\sin\rho.$$

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Isosurfaces For Regular Basis Functions

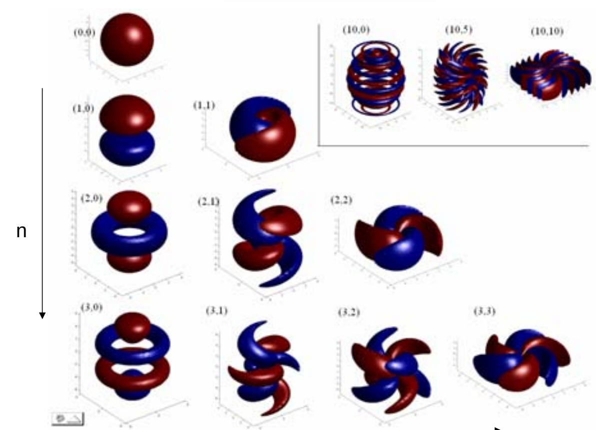
$$\mathrm{Re}\{R_n^m(\mathbf{r})\} = const$$



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Isosurfaces For Singular Basis Functions

$$\mathrm{Re}\{S_n^m(\mathbf{r})\} = const$$



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Expansions

$$\psi(\mathbf{r}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} A_n^m F_n^m(\mathbf{r}) = \sum_{m=-\infty}^{\infty}\sum_{n=|m|}^{\infty} A_n^m F_n^m(\mathbf{r}), \quad F = S, R, \quad A_n^m \in \mathbb{C}.$$

Absolute and uniform convergence

$$\forall\epsilon > 0, \quad \exists p(\epsilon), \quad \left|\psi(\mathbf{r}) - \sum_{n=0}^{p-1}\sum_{m=-n}^{n} A_n^m F_n^m(\mathbf{r})\right| < \epsilon, \quad \forall \mathbf{r} \in \Omega,$$

and

$$\forall\epsilon > 0, \quad \exists p(\epsilon), \quad \sum_{n=p}^{\infty}\sum_{m=-n}^{n} |A_n^m F_n^m(\mathbf{r})| < \epsilon, \quad \forall \mathbf{r} \in \Omega.$$

Plane Wave expansion:

$$e^{i\mathbf{k}\cdot\mathbf{r}} = 4\pi \sum_{n=0}^{\infty}\sum_{m=-n}^{n} i^n Y_n^{-m}(\theta_k, \varphi_k) R_n^m(\mathbf{r}),$$

$$\mathbf{k} = k\mathbf{s}, \quad \mathbf{s} = (\sin\theta_k\cos\varphi_k, \sin\theta_k\sin\varphi_k, \cos\theta_k).$$

Wave vector

Matrix Translations

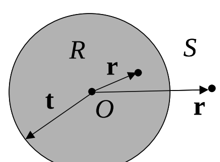
Reexpansions of Basis Functions

$$R_n^m(\mathbf{r}+\mathbf{t}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (R|R)_{n'n}^{m'm}(\mathbf{t}) R_{n'}^{m'}(\mathbf{r}), \quad n = 0,1,2,\ldots, \quad m = -n,\ldots,n.$$

Reexpansion Matrices

$$S_n^m(\mathbf{r}+\mathbf{t}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} \begin{cases} (S|R)_{n'n}^{m'm}(\mathbf{t}) R_{n'}^{m'}(\mathbf{r}), & |\mathbf{r}| < |\mathbf{t}| \\ (S|S)_{n'n}^{m'm}(\mathbf{t}) S_{n'}^{m'}(\mathbf{r}), & |\mathbf{r}| > |\mathbf{t}| \end{cases},$$

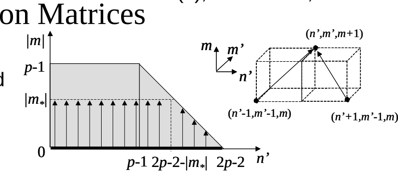
$$n = 0,1,2,\ldots, \quad m = -n,\ldots,n.$$

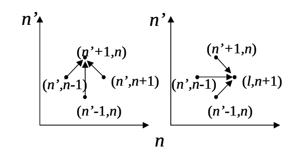


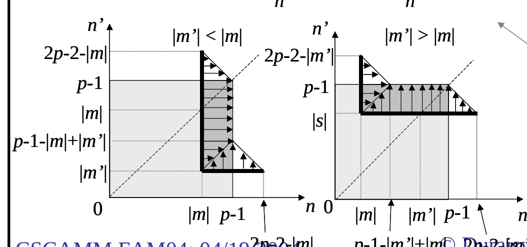
Recursive Computation of Reexpansion Matrices

Gumerov & Duraiswami, *SIAM J. Sci. Stat. Comput.* 25(4), 1344-1381, 2003.

p^4 elements of the truncated reexpansion matrices can be computed for $O(p^4)$ operations recursively







$(E|F)_{n'|m|}^{m'm}$

$(E|F)_{n'n}^{m'm}$

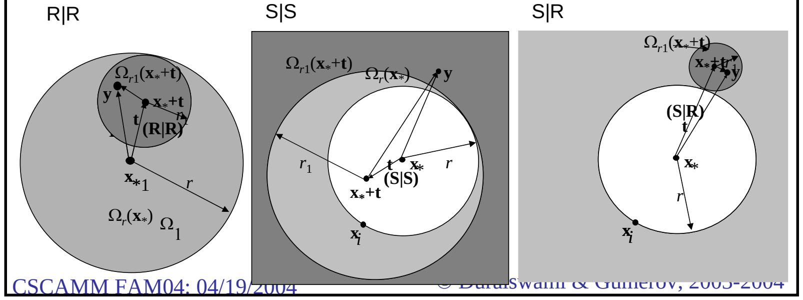
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Translations

$$\psi(\mathbf{y}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} C_n^m(\mathbf{x}_{*1}) E_n^m(\mathbf{y}-\mathbf{x}_{*1}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} C_n^m(\mathbf{x}_{*2}) F_n^m(\mathbf{y}-\mathbf{x}_{*2}),$$

$$C_n^m(\mathbf{x}_{*2}) = \sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'} (E|F)_{n'n}^{m'm}(\mathbf{t}) C_{n'}^{m'}(\mathbf{x}_{*1}), \quad \mathbf{t} = \mathbf{x}_{*2} - \mathbf{x}_{*1}$$

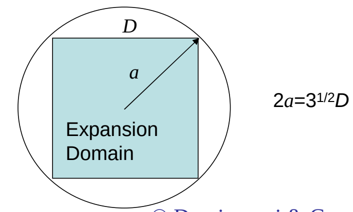
$$E,F = S,R, \quad n = 0,1,\ldots, \quad m = -n,\ldots,n.$$



Problem:

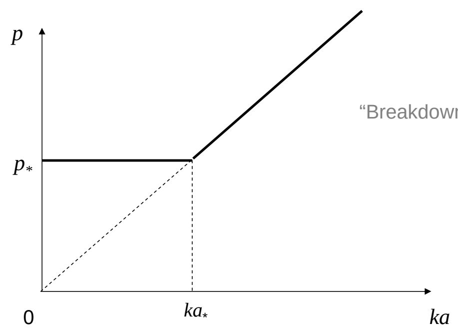
- For the Helmholtz equation absolute and uniform convergence can be achieved only for

  $p > ka$. For large ka the FMM with constant p is
  - very expensive (comparable with straightforward methods);
  - inaccurate (since keeps much larger number of terms than required, which causes numerical instabilities).



$2a=3^{1/2}D$

Model of Truncation Number Behavior for Fixed Error



In the multilevel FMM we associate its own p_l with each level l:

$$P_l = p_l^2$$

"Breakdown level"

$$kD_{l_{b*}} = \kappa_b, \quad l_{b*} = \log_2 \frac{kD_0}{\kappa_b}$$

$$P_l = A(kD_l)^2, \quad l \leqslant l_{b*}$$

$$P_l = A\kappa_b^2 = P_*, \quad l > l_{b*}$$

Box size at level l

Complexity of Single Translation

Translation exponent

$$CostTrans(P_l) = CP_l^{\nu} = Cp_l^{2\nu}, \quad l = 2,\ldots,l_{\max}.$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Spatially Uniform Data Distributions

$$N_l \sim 8^{-l}N, \quad l_{\max} \sim \frac{1}{3}\log N$$

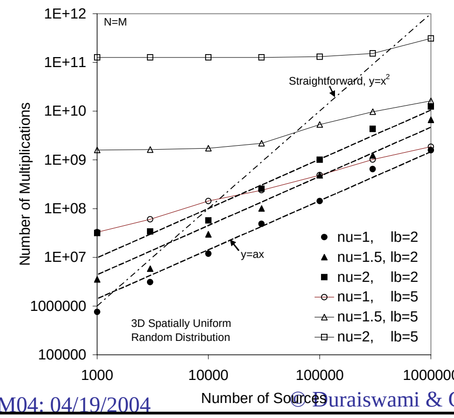
$$p_l \sim 2^{-l}kD_0,$$

$$N_{oper} \sim (kD_0)^{2\nu}\sum_{l=2}^{l_{\max}} 2^{-2\nu l}8^l = (kD_0)^{2\nu}\sum_{l=2}^{l_{\max}} 2^{(3-2\nu)l}.$$

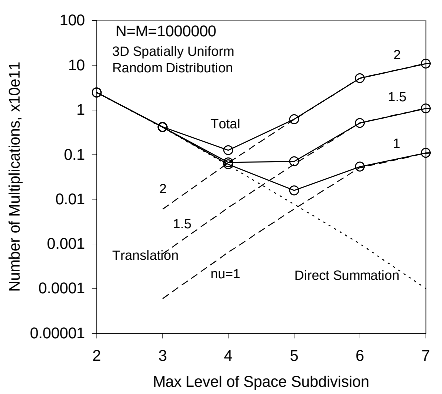
- $\nu < 1.5: \quad ComplexityFMM \sim (kD_0)^{2\nu}2^{(3-2\nu)l_{\max}} \sim (kD_0)^{2\nu}N^{1-2\nu/3}$
- $\nu = 1.5: \quad ComplexityFMM \sim (kD_0)^{2\nu}l_{\max} \sim (kD_0)^{2\nu}\log N$
- $\nu > 1.5: \quad ComplexityFMM \sim (kD_0)^{2\nu}$

Constant!

Complexity of the Optimized FMM for Fixed kD_0 and Variable N



Optimum Level for Low Frequencies



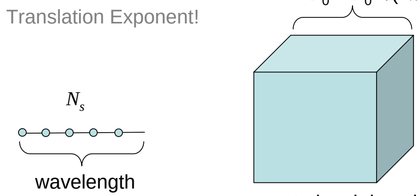
Volume Element Methods

$$N = \left(\frac{N_s}{2\pi}kD_0\right)^3, \quad kD_0 \sim N^{1/3}$$

- $\nu < 1.5: \quad ComplexityFMM \sim (kD_0)^{2\nu}2^{(3-2\nu)l_{\max}} \sim (kD_0)^{2\nu}N^{1-2\nu/3} \sim N$
- $\nu = 1.5: \quad ComplexityFMM \sim (kD_0)^{2\nu}l_{\max} \sim (kD_0)^{2\nu}\log N \sim N\log N$
- $\nu > 1.5: \quad ComplexityFMM \sim (kD_0)^{2\nu} \sim N^{2\nu/3} \gg N\log N.$

Critical Translation Exponent!

$D_0 = D_0\, k/(2\pi)$ wavelengths = $N^{1/3}$ sources



gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

What Happens if Truncation Number is Constant for All Levels?

$$N_{oper} \sim (kD_0)^{2\nu} \sum_{l=2}^{l_{\max}} 8^l = (kD_0)^{2\nu} \sum_{l=2}^{l_{\max}} 2^{3l} \sim (kD_0)^{2\nu} 2^{3l_{\max}} \sim (kD_0)^{2\nu} N \sim N^{1+2\nu/3}.$$

- $\nu < 1.5: \quad N \ll ComplexityFMM \ll N^2$
- $\nu = 1.5: \quad ComplexityFMM \sim N^2$
- $\nu > 1.5: \quad ComplexityFMM \sim N^{1+2\nu/3} \gg N^2$

"Catastrophic Disaster of the FMM"

Surface Data Distributions

$$N_l \sim 4^{-l} N, \quad l_{\max} \sim \frac{1}{2}\log N$$
$$p_l \sim 2^{-l} kD_0,$$

$$N_{oper} \sim (kD_0)^{2\nu} \sum_{l=2}^{l_{\max}} 2^{-2\nu l} 4^l = (kD_0)^{2\nu} \sum_{l=2}^{l_{\max}} 2^{(2-2\nu)l}.$$

- $\nu = 1: \quad ComplexityFMM \sim (kD_0)^{2\nu} l_{\max} \sim (kD_0)^{2\nu} \log N$
- $\nu > 1: \quad ComplexityFMM \sim (kD_0)^{2\nu}$

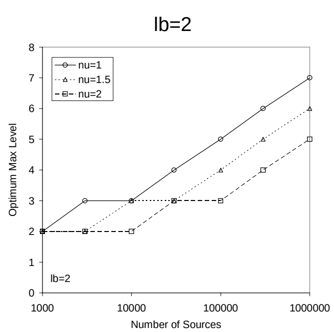
Boundary Element Methods:

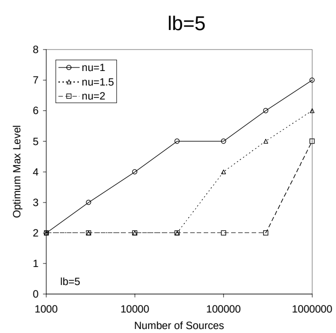
$$N = \left(\frac{N_s}{2\pi} kD_0\right)^2, \quad kD_0 \sim N^{1/2}$$

- $\nu = 1: \quad ComplexityFMM \sim (kD_0)^{2\nu} l_{\max} \sim (kD_0)^{2\nu} \log N \sim N \log N$
- $\nu > 1: \quad ComplexityFMM \sim (kD_0)^{2\nu} \sim N^{\nu} \gg N \log N.$

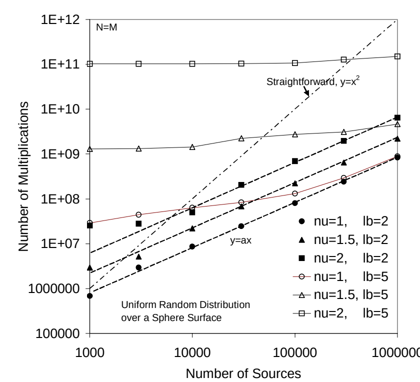
Critical Translation Exponent!

Optimum Level of Space Subdivision





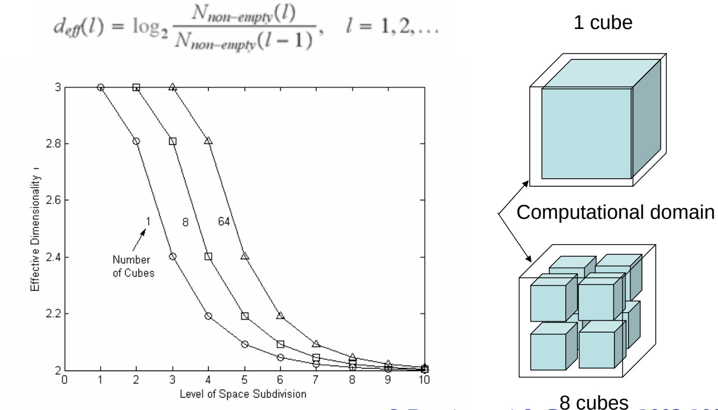
Performance of the MLFMM for Surface Data Distributions



gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Effective Dimensionality of the Problem

$$d_{eff}(l) = \log_2 \frac{N_{non-empty}(l)}{N_{non-empty}(l-1)}, \quad l = 1, 2, \ldots$$



Fast Translation Methods

Translation Methods

- $O(p^5)$: Matrix Translation with Computation of Matrix Elements Based on Clebsch-Gordan Coefficients;
- $O(p^4)$ (Low Asymptotic Constant): Matrix Translation with Recursive Computation of Matrix Elements
- $O(p^3)$ (Low Asymptotic Constants):
  - Rotation-Coaxial Translation Decomposition with Recursive Computation of Matrix Elements;
  - Sparse Matrix Decomposition;
- $O(p^2 \log^\beta p)$
  - Rotation-Coaxial Translation Decomposition with Structured Matrices for Rotation and Fast Legendre Transform for Coaxial Translation;
  - Translation Matrix Diagonalization with Fast Spherical Transform;
  - Asymptotic Methods;
  - Diagonal Forms of Translation Operators with Spherical Filtering.

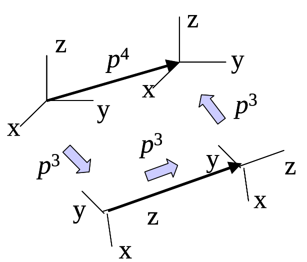
$O(p^3)$ Methods

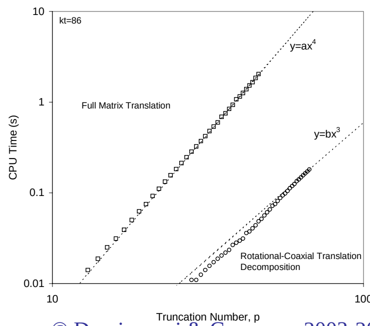
gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Rotation - Coaxial Translation Decomposition (Complexity O(p^3))

From the group theory follows that general translation can be reduced to

$$(\mathbf{F}|\mathbf{E})(\mathbf{t}) = \mathbf{Rot}(Q^{-1})(\mathbf{F}|\mathbf{E})_{(coax)}(t)\mathbf{Rot}(Q), \quad F,E = S,R.$$





Sparse Matrix Decomposition

$$(\mathbf{R}|\mathbf{R})(\mathbf{t}) = (\mathbf{S}|\mathbf{S})(\mathbf{t}) = \sum_{n=0}^{\infty} \frac{(kt)^n}{n!}\mathbf{D}_\mathbf{t}^n = e^{kt\mathbf{D}_\mathbf{t}} = \Lambda_r(kt, -i\mathbf{D}_\mathbf{t})$$

$$(\mathbf{S}|\mathbf{R})(\mathbf{t}) = \Lambda_s(kt, -i\mathbf{D}_\mathbf{t})$$

$$\Lambda_r(kt, -i\mathbf{D}_\mathbf{t}) = \sum_{n=0}^{\infty}(2n+1)i^n j_n(kt)P_n(-i\mathbf{D}_\mathbf{t})$$

$$\Lambda_s(kt, -i\mathbf{D}_\mathbf{t}) = \sum_{n=0}^{\infty}(2n+1)i^n h_n(kt)P_n(-i\mathbf{D}_\mathbf{t}).$$

Matrix-vector products with these matrices computed recursively

$$(\mathbf{D}_\mathbf{t}\mathbf{C})_n^m = \frac{1}{2t}\left[(t_x + it_y)\left(C_{n-1}^{m+1}b_n^m - C_{n+1}^{m+1}b_{n+1}^{-m-1}\right) + (t_x - it_y)\left(C_{n-1}^{m-1}b_n^{-m} - C_{n+1}^{m-1}b_{n+1}^{m-1}\right)\right]$$
$$+ \frac{t_z}{t}(a_n^m C_{n+1}^m - a_{n-1}^m C_{n-1}^m), \qquad m = 0, \pm1, \pm2, \ldots, \quad n = |m|, |m|+1, \ldots$$

O(p^{2}log$^\beta$p) Methods

Fast Rotation Transform

Diagonal matrices

$$\mathbf{Rot}(Q(\alpha,\beta,\gamma)) = \mathbf{Rot}\left(A\left(\gamma + \frac{\pi}{2}\right)\right)\mathbf{Rot}\left(B\left(\frac{\pi}{2}\right)\right)\mathbf{Rot}(A(\beta))\mathbf{Rot}\left(B\left(\frac{\pi}{2}\right)\right)\mathbf{Rot}\left(A\left(\frac{\pi}{2} - \alpha\right)\right).$$

Euler angles

Block-Toeplitz matrices

$$\mathbf{Rot}\left(B\left(\frac{\pi}{2}\right)\right) = \mathbf{LK}^T\mathbf{MKL}$$

Diagonal matrices

Complexity: O(p^{2}logp)

Presented at the Center for Scientific Computing and Mathematical Modeling, University of Maryland, College Park

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Fast Coaxial Translation

$$(\mathbf{R}|\mathbf{R})^{(p,p')}_{(coax)}(t) = (\mathbf{S}|\mathbf{S})^{(p,p')}_{(coax)}(t) = \mathbf{i}^{(p)}\underline{\mathbf{L}}^{(p)}\mathbf{W}\Lambda_r^{(p+p'-1)}(kt)\left(\underline{\mathbf{L}}^{(p')}\right)^T\overline{\mathbf{i}^{(p')}},$$

$$(\mathbf{S}|\mathbf{R})^{(p,p')}_{(coax)}(t) = \mathbf{i}^{(p)}\underline{\mathbf{L}}^{(p)}\mathbf{W}\Lambda_s^{(p+p'-1)}(kt)\left(\underline{\mathbf{L}}^{(p')}\right)^T\overline{\mathbf{i}^{(p')}}.$$

Legendre and transposed Legendre matrices

Diagonal matrices

Fast multiplication of the Legendre and transposed Legendre matrices can be performed via the forward and inverse FAST LEGENDRE TRANSFORM (FLT) with complexity $O(p^2 \log^2 p)$

Healy et al *Advances in Computational Mathematics* **21**: 59-105, 2004.

Diagonalization of General Translation Operator

$$(\mathbf{E}|\mathbf{F})^{(p,p')}(\mathbf{t}) = \mathbf{i}^{(p)}\mathbf{Y}^{(p)}\mathbf{W}\Lambda^{(p+p'-1)}(\mathbf{t})\left(\overline{\mathbf{Y}^{(p')}}\right)^T\overline{\mathbf{i}^{(p')}}.$$

Matrices for the forward and inverse and Spherical Transform

Diagonal matrices

FAST SPHERICAL TRANSFORM (FST) can be performed with complexity $O(p^2 \log^2 p)$

Healy et al *Advances in Computational Mathematics* **21**: 59-105, 2004.

Method of Signature Function (Diagonal Forms of the Translation Operator)

$$\psi(\mathbf{r}) = \frac{1}{4\pi}\int_{S_u} e^{ik\mathbf{s}\cdot\mathbf{r}}\Psi(\mathbf{s})dS(\mathbf{s}),$$ Regular Solution

$$\psi^{(p)}(\mathbf{r}) = \frac{1}{4\pi}\int_{S_u} \Lambda_s^{(p)}(\mathbf{r};\mathbf{s})\Psi(\mathbf{s})dS(\mathbf{s}),$$ Singular Solution

$$\Lambda_r(\mathbf{r};\mathbf{s}) = \sum_{n=0}^{\infty}(2n+1)i^n j_n(kr)P_n\left(\frac{\mathbf{r}\cdot\mathbf{s}}{r}\right)$$

$$\Lambda_s^{(p)}(\mathbf{r};\mathbf{s}) = \sum_{n=0}^{p-1}(2n+1)i^n h_n(kr)P_n\left(\frac{\mathbf{r}\cdot\mathbf{s}}{r}\right).$$

$$\widehat{\Psi}(\mathbf{s}) = (\mathcal{S}|\mathcal{S})(\mathbf{t})[\Psi(\mathbf{s})] = (\mathcal{R}|\mathcal{R})(\mathbf{t})[\Psi(\mathbf{s})] = e^{ik\mathbf{s}\cdot\mathbf{t}}\Psi(\mathbf{s}),$$

$$\widehat{\Psi}_{(p)}(\mathbf{s}) = (\mathcal{S}|\mathcal{R})(\mathbf{t})[\Psi(\mathbf{s})] = \Lambda_s^{(p)}(\mathbf{t};\mathbf{s})\Psi(\mathbf{s}).$$

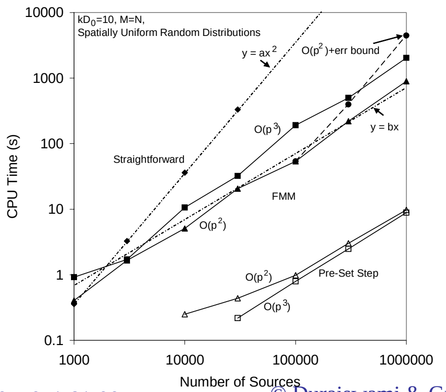
Final Summation and Initial Expansion

$$\psi(\mathbf{r}) = \frac{1}{4\pi}\sum_{j=0}^{N_c-1} w_j e^{ik\mathbf{s}_j\cdot\mathbf{r}}\Psi(\mathbf{s}_j) + \epsilon_c, \quad \mathbf{s}_j \in S_u,$$

$$G(\mathbf{r}-\mathbf{r}_s) \rightleftarrows \Psi_{(0)}(\mathbf{s}_j;\mathbf{r}_s-\mathbf{r}_*) = \frac{ik}{4\pi}e^{-ik\mathbf{s}_j\cdot(\mathbf{r}_c-\mathbf{r}_*)}$$

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

The FMM with Band-Unlimited Signature Functions ($O(p^2)$ method)



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Deficiencies

- Low Frequencies;
- High Frequencies;
- Constant p;
- Instabilities after two or three levels of translations.

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

Methods to Fix:

- Use of Band-limited functions;
- Error control via band-limits;
- Requires filtering procedures (complexity $O(p^2\log^2 p)$ or $O(p^2\log p)$) with large asymptotic constants;
- The length of the representation is changed via interpolation/anterpolation procedures.

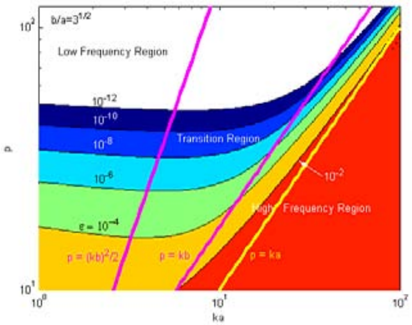
CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

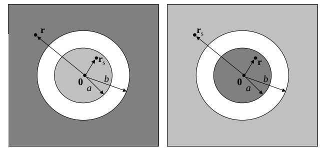
Error Bounds

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

Source Expansion Errors





Low frequencies:

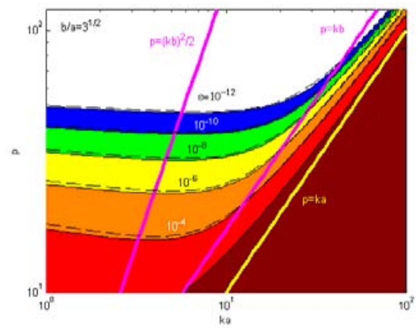
$$p = -\frac{\ln\left[\epsilon ka(1-\sigma^{-1})^{3/2}\right]}{\ln\sigma} - 1.$$

High frequencies:

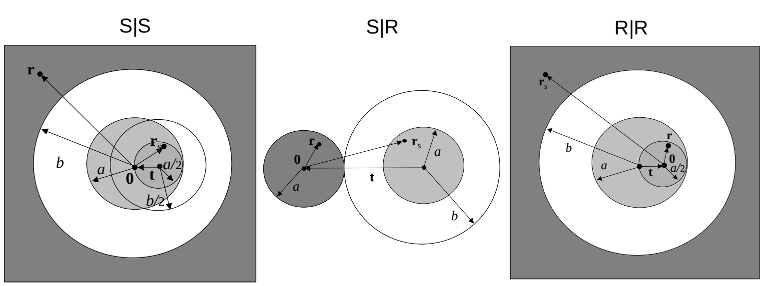
$$p = ka + \frac{1}{2}\left(3\ln\frac{1}{\epsilon\sigma}\right)^{2/3}(ka)^{1/3}.$$

Approximation of the Error

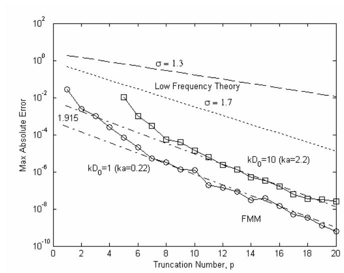
$$p = \left\{\left[\frac{1}{\ln\sigma}\ln\frac{1}{\epsilon ka(1-\sigma^{-1})^{3/2}} + 1\right]^4 + \left[ka + \frac{1}{2}\left(3\ln\frac{1}{\epsilon\sigma}\right)^{2/3}(ka)^{1/3}\right]^4\right\}^{1/4}$$

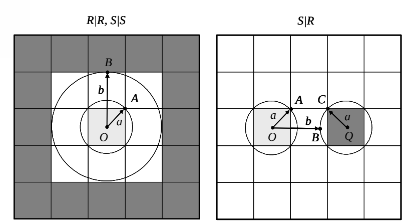


We proved that for source summation problems the truncation numbers can be selected based on the above chart when using translations with rectangularly truncated matrices



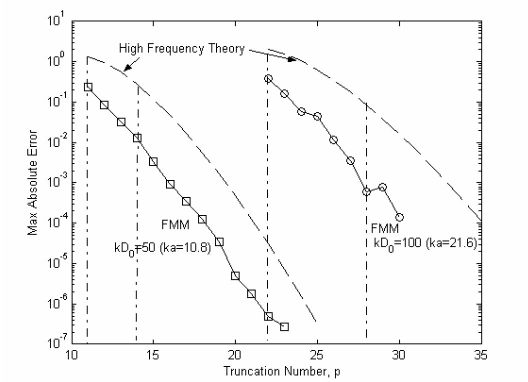
Low Frequency FMM Error





gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

High Frequency FMM Error



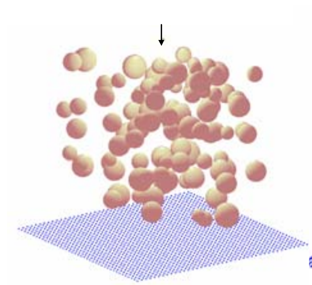
Multiple Scattering Problem

CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

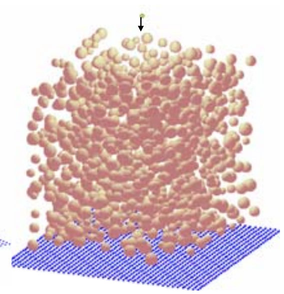
Problem

Boundary Conditions:

$$|\mathbf{r}-\mathbf{r}'_q| = a_q: \quad \frac{\partial\psi(\mathbf{r})}{\partial n_q} + i\sigma_q\psi(\mathbf{r}) = 0, \quad q = 1,\ldots,N.$$



100 random spheres



1000 random spheres

CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

T-Matrix Method

Scattered Field Decomposition

$$\psi_{scat}(\mathbf{r}) = \sum_{p=1}^{N}\psi_p(\mathbf{r}), \quad \lim_{r\to\infty} r\left(\frac{\partial\psi_p}{\partial r} - ik\psi_p\right) = 0, \qquad p = 1,\ldots,N.$$

Singular Basis Functions

Hankel Functions

$$\psi_p(\mathbf{r}) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} A_n^{(p)m} S_n^m(\mathbf{r}-\mathbf{r}'_p), \qquad S_n^m(\mathbf{r}) = h_n(kr)Y_n^m(\theta,\varphi).$$

Expansion Coefficients

Spherical Harmonics

$$\mathbf{A} = \left(A_0^0, A_1^{-1}, A_1^0, A_1^1, A_2^{-2}, A_2^{-1}, A_2^0, A_2^1, A_2^2, \ldots\right)^T,$$

Vector Form:

$$\psi_p(\mathbf{r}) = \overline{\mathbf{A}^{(p)}} \cdot \mathbf{S}(\mathbf{r}-\mathbf{r}'_p).$$

dot product

CSCAMM FAM04: 04/19/2004
© Duraiswami & Gumerov, 2003-2004

gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

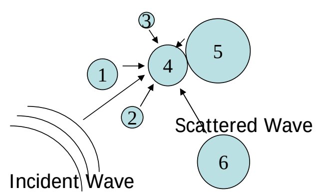
Solution of Multiple Scattering Problem

"Effective" Incident Field

$$\psi(\mathbf{r}) = \psi_q(\mathbf{r}) + \psi_{in}(\mathbf{r}) + \psi^{(q)}_{other}(\mathbf{r}) = \psi_q(\mathbf{r}) + \psi^{(q)(in)}_{eff}(\mathbf{r}),$$

$$\psi^{(q)}_{other}(\mathbf{r}) = \sum_{p \neq q} \overline{\mathbf{A}^{(p)}} \cdot \mathbf{S}(\mathbf{r}_p) = \overline{\mathbf{B}^{(q)}} \cdot \mathbf{R}(\mathbf{r}_q), \quad \psi^{(q)(in)}_{eff}(\mathbf{r}) = \overline{\mathbf{E}^{(q)}_{eff}} \cdot \mathbf{R}(\mathbf{r}_q).$$

Coupled System of Equations:



$$\mathbf{A}^{(q)} = \mathbf{T}^{(q)}\mathbf{E}^{(q)}_{eff},$$

$$\mathbf{B}^{(q)} = \sum_{p \neq q} (\mathbf{S}|\mathbf{R})(\mathbf{r}'_q - \mathbf{r}'_p)\mathbf{A}^{(p)},$$

$$\mathbf{E}^{(q)}_{eff} = \mathbf{E}^{(in)}(\mathbf{r}'_q) + \mathbf{B}^{(q)},$$

$$q = 1, \ldots, N$$

(S|R)-Translation Matrix

Reflection Method & Krylov Subspace Method (GMRES)

Reflection (Simple Iteration) Method:

$$\mathbf{A}^{(q)}_j = \mathbf{T}^{(q)}\left[\mathbf{E}^{(in)}(\mathbf{r}'_q) + \mathbf{B}^{(q)}_j\right],$$

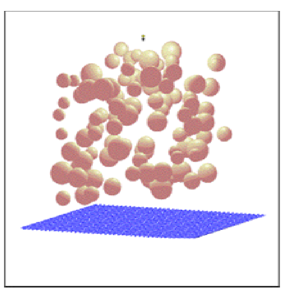
$$\mathbf{B}^{(q)}_{j+1} = \sum_{p \neq q} (\mathbf{S}|\mathbf{R})(\mathbf{r}'_q - \mathbf{r}'_p)\mathbf{A}^{(p)}_j,$$

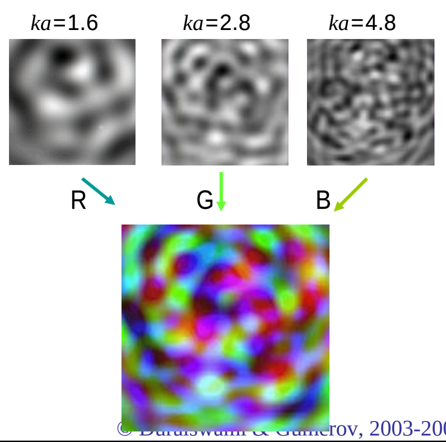
$$\left|\mathbf{A}^{(q)}_j - \mathbf{A}^{(q)}_{j+1}\right| < \epsilon, \quad q = 1, \ldots, N.$$

General Formulation (used in GMRES)

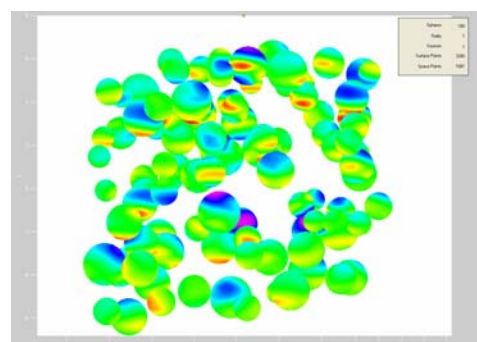
$$\left[\mathbf{I} - \mathbf{T}^{(q)} \sum_{p \neq q} (\mathbf{S}|\mathbf{R})(\mathbf{r}'_q - \mathbf{r}'_p)\right]\mathbf{A}^{(q)} = \mathbf{T}^{(q)}\mathbf{E}^{(in)}(\mathbf{r}'_q).$$

100 random spheres (MLFMM)



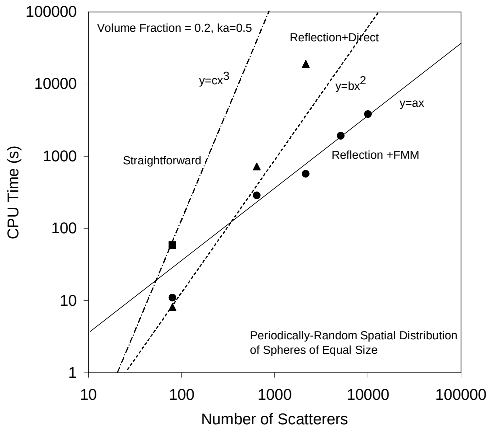


Surface Potential Imaging



gumerov@umiacs.umd.edu
ramani@umiacs.umd.edu

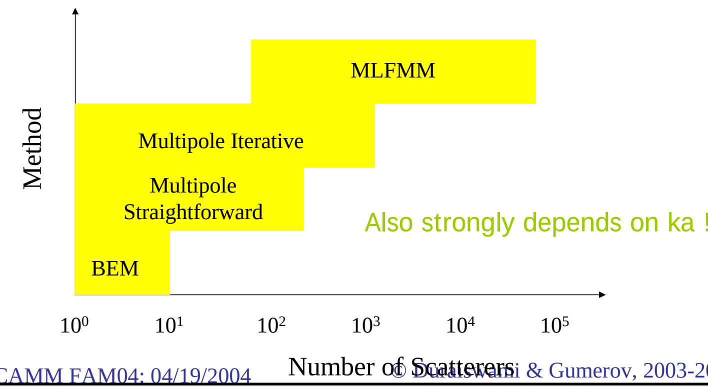
Performance Test



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

MLFMM

Computable Problems on Desktop PC



CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004

More About This Problem in Our Talk Next Week

CSCAMM FAM04: 04/19/2004 © Duraiswami & Gumerov, 2003-2004