Outline

• Take home message
  – Recognizing structure and exploiting it saves time

• Research Problems and Open Questions
  – Representations and Basis Functions
  – Space adaptivity
  – $P$ adaptivity
  – Exact Fast translation
  – Kronecker Products
  – Approximate Fast translation
  – Data structures

• Review of the Course
Structure

- Structured matrix
  - Number of elements is much less than $N^2$
- Fourier: $O(1)$ parameters
- Toeplitz, Hankel, etc. on $O(N)$ parameters
- FMM on $O(N)$ parameters
- Recently there have been some problems for which there are sublinear algorithms
  - FFT when the signal is known to have very few frequencies
General view of basis functions

• In general basis functions are obtained from solutions of PDEs

• Translation and other identities are obtained from handbooks or derived

• However, it may be possible to think of a general theory of translation of RBFs from symmetry and other considerations

• Outline a preliminary approach here
General view of Basis functions in the FMM

• Goal of FMM is evaluation of
  \[ v(y) = \sum_{k=0}^{N} u_k \Phi(y, x_k), \]

• Let \( \{x_i\} \) and \( \{y_j\} \) be point sets \( \in \mathbb{R}^d \)

• Consider the “Mother function” \( \Phi(x, y) \)
  – usually singular
  – Usually a radial function
  – It depends only on the distance between \( x \) and \( y \)

• Properties
  \[ \Phi(y, x_k) = \phi(|y - x_k|), \]
  – Symmetry
  \[ \Phi(y, x_k) = \Phi(x_k, y) \]
  – Local rotations do not change value
  \[ \Phi(y, x_k) = \Phi(Q_1 x_k, Q_2 y) \]
Duality of the Basis and Coefficients

Consider expansion of this function into series

\[ \Phi(y, x_k) = \sum_{m=0}^{\infty} a_m(x_k - x_{*1})R_m(y - x_{*1}), \quad |y - x_{*1}| < |x_k - x_{*1}|, \]

\[ \Phi(y, x_k) = \sum_{m=0}^{\infty} b_m(x_k - x_{*1})S_m(y - x_{*1}), \quad |y - x_{*1}| > |x_k - x_{*1}|. \]

Indeed,

\[ \Phi(y, x_k) = \phi(|y - x_k|) = \phi(|y - x_{*1} - (x_k - x_{*1})|), \]

so coefficients should depend on differences \((x_k - x_{*1})\) only. Due to symmetry we have

\[ \Phi(x_k, y) = \sum_{m=0}^{\infty} a_m(y - x_{*1})R_m(x_k - x_{*1}), \quad |y - x_{*1}| > |x_k - x_{*1}|, \]

\[ \Phi(x_k, y) = \sum_{m=0}^{\infty} b_m(y - x_{*1})S_m(x_k - x_{*1}), \quad |y - x_{*1}| < |x_k - x_{*1}|. \]

Can Prove

\[ a_m(y) = S_m(y), \]

\[ b_m(y) = R_m(y). \]
For this purpose consider case \( y = x_{*1} \). So we have

\[
\Phi(x_{*1}, x_k) = \sum_{m=0}^{\infty} a_m(x_k - x_{*1})R_m(0), \quad |x_k - x_{*1}| > 0,
\]

\[
\Phi(x_k, x_{*1}) = \sum_{m=0}^{\infty} b_m(0)S_m(x_k - x_{*1}), \quad |x_k - x_{*1}| > 0.
\]

We can match both expansions in region \( |x_k - x_{*1}| > 0 \), by setting

\[
b_m(0) = R_m(0), \quad a_m(x_k - x_{*1}) = S_m(x_k - x_{*1}).
\]

So

\[
\Phi(y, x_k) = \sum_{m=0}^{\infty} S_m(x_k - x_{*1})R_m(y - x_{*1}), \quad |y - x_{*1}| < |x_k - x_{*1}|.
\]
Properties of basis functions

- We require

  Basis functions of order 0
  
  Now we note that the basis can be selected as follows:

  \[ R_0(0) = 1, \]
  \[ R_m(0) = 0, \quad m > 0. \]
  \[ S_0(y) = \Phi(y, 0) = \phi(|y|). \]

- We can then view the FMM as requiring two sets of expansions

  \[
  \Phi(y, x_k) = \phi(|y|) = S_0(y - x_k) = \sum_{m=0}^{\infty} S_m(x_k - x_{k+1}) R_m(y - x_{k+1}), \quad |y - x_{k+1}| < |x_k - x_{k+1}|
  \]

  \[
  \Phi(y, x_k) = \phi(|y|) = S_0(y - x_k) = \sum_{m=0}^{\infty} R_m(x_k - x_{k+1}) S_m(y - x_{k+1}), \quad |y - x_{k+1}| > |x_k - x_{k+1}|
  \]

- Monomials are a basis that can be used for \( R \) expansions
- This leads to Taylor series for \( R \) and Laurent for \( S \)
General view of basis functions

- Using Taylor series leads to evaluation of multiple derivatives of the “mother function”
- Can use any basis
- In general FMM used in the solution of PDEs
- Eigenfunctions of these equations are usually used as basis functions
- Because functions satisfy the differential equation we have $d-1$ summations as opposed to $d$ summations.
- Research Questions:
  - Why does this trick work for some functions and not for others?
  - Or, if it works for all, how do we find the right basis?
S- and R- expansions of Fundamental Solution

\[
\frac{1}{|r - r_0|} = \frac{4\pi}{r_0} \sum_{n=0}^{\infty} \frac{1}{2n + 1} \left( \frac{r}{r_0} \right)^n \sum_{m=-n}^{n} Y_n^{-m}(\theta', \varphi') Y_n^m(\hat{\theta}, \hat{\varphi}), \quad r < r_0,
\]

\[
\frac{1}{|r - r_0|} = \frac{4\pi}{r_0} \sum_{n=0}^{\infty} \frac{1}{2n + 1} \left( \frac{r_0}{r} \right)^{n+1} \sum_{m=-n}^{n} Y_n^{-m}(\theta', \varphi') Y_n^m(\hat{\theta}, \hat{\varphi}), \quad r > r_0.
\]

\[
r = r(\sin \theta \cos \varphi, \sin \theta \sin \varphi, \cos \theta),
\]

\[
R_n^m(r) = r^n Y_n^m(\theta, \varphi),
\]

\[
S_n^m(r) = r^{-n-1} Y_n^m(\hat{\theta}, \hat{\varphi}),
\]

\[
\frac{1}{4\pi|r - r_0|} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{1}{2n + 1} S_n^{-m}(r_0) R_n^m(r), \quad r < r_0,
\]

\[
\frac{1}{4\pi|r - r_0|} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{1}{2n + 1} R_n^{-m}(r_0) S_n^m(r), \quad r > r_0.
\]

Multipole (!)
Product form and associated instability of $A \cdot B$

From previous Lectures …

$$S_n(y-x_*+t) = (t+y)^{-n-1} = \sum_{m=0}^{\infty} \frac{1}{m!} \frac{d^m S_n(t)}{dt^m} (y-x_*)^m$$

$$= \sum_{m=0}^{\infty} \frac{1}{m!} \frac{d^m S_n(t)}{dt^m} R_m(y-x_*) = \sum_{m=0}^{\infty} (S|R)_{mn}(t) R_m(y-x_*).$$

So

$$(S|R)_{mn}(t) = \frac{1}{m!} \frac{d^m S_n(t)}{dt^m} = \frac{(-1)^m (m+n)!}{m! n! t^{n+m+1}}.$$
Instability … (contd)

• So for small or large values $t$ or $y-x_*$ we have large and small numbers multiplying each other.

• FMM factorizes these

• Two problems
  – Explosive growth of a quantity that is finite before factorization
  – Loss of precision in finite-precision arithmetic

• Open research problem
  – Device techniques to “balance” the factorization
  – Can involve either scaling functions
  – Alternatively we can multiply by a growing function $f(k)$ in the collapsing part, and divide by it in the growing part.
Scheme of the FMM

Source Data Hierarchy

N

MLFMM

SIS

Level 2

Level 3

Level 5

Level 4

Evaluation Data Hierarchy

M

SIR

RIR

Level 2

Level 3

Level 4

Level 5
Review

- Data Structured into boxes containing $s$ particles
- $S$ expansion about box level at finest center

\[ |y - x_*| > |x_i - x_*| : \Phi(y, x_i) = \sum_{m=0}^{\infty} b_m(x_i, x_*) S_m(y - x_*), \]

- Move up source tree
- $S|R$ expansion to allowable boxes at highest level
- $R|R$ translation and further $S|R$ translations

\[ |y - x_*| < |x_i - x_*| : \Phi(y, x_i) = \sum_{m=0}^{\infty} a_m(x_i, x_*) R_m(y - x_*), \]

- Final evaluation and Direct summation of unallowable
Hierarchical Spatial Domains

$E_1$

$E_2$

$E_3$

$E_4$
Cost of MLFMM

Regular mesh: \( N = 2^{L \ast d}, \quad s = 2^{L_s d}, \quad L = L_{\text{max}} = L_{\ast} - L_s. \)

\[ \text{CostUpward}_1 = N \text{CostExpansion}(P) = O(NP). \]

\[ \text{CostUpward}_2 = 2^d \left( 2^{(L-1)d} + 2^{(L-2)d} + \ldots + 2^{2d} \right) \text{CostSS}(P) \]

\[ \leq \frac{2^d}{2^d - 1} \left( 2^{L_d} - 1 \right) \text{CostSS}(P) \sim \frac{N}{s} \text{CostSS}(P) \]

\[ \text{CostDownward}_1 \leq P_4(d) \left( 2^{2d} + \ldots + 2^{L_d} \right) \text{CostSR}(P) \sim P_4(d) \frac{N}{s} \text{CostSR}(P), \]

\[ \text{CostDownward}_2 = 2^d \left( 2^{2d} + \ldots + 2^{(L-1)d} \right) \text{CostRR}(P) \sim \frac{N}{s} \text{CostRR}(P), \]

\[ \text{CostEvaluation} = M(P_2(d) s \text{CostFunc} + P). \]

Assume that all translation costs are the same, \( \text{CostTranslation}(P) \)

\[ \text{CostMLFMM} = (M + N) P + (P_4(d) + 2) \frac{N}{s} \text{CostTranslation}(P) + P_2(d) s M. \]
Downward Pass. Step 1.

\[ P_4 = \text{PowerOfE}_4\text{Neighborhood} = 3^d2^d - 3^d = 3^d(2^d - 1) \]

For different dimensions:

\begin{align*}
  d = 1 : & \quad P_4 = 3, \\
  d = 2 : & \quad P_4 = 27, \\
  d = 3 : & \quad P_4 = 189, \\
  d = 4 : & \quad P_4 = 1215, \\
  \ldots
\end{align*}

Exponential Growth

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

(far from the domain boundaries)
Main Cost: S|R translations

- S|R translations also increase the value of $P$
- Why?
  - For some $\Phi$ they are evaluated in the least favorable situations
  - $E_4$ neighborhood
- Direct Matrix vector product method for calculating translation requires $O(P^2)$ operations
- For slowly convergent series $P$ can be quite large
- Way to attack problem
  - Avoid Calculations you don’t need
  - Improve speed of coefficient computation
  - Improve speed of translation
Data structures for the FMM

- FMM always implemented with the box data structure
- Could implement it with other space partitioning schemes
- Why do we need a data structure?
  - To ensure that we are able to partition points cheaply into boxes that are “well-separated”
- Boxes are not optimal as the dimensions increase.
  - Ratio of circumsphere to box volume
- Yang et al, implemented the Fast Gauss Transform using k-center clustering
Adaptive Clustering

• Discussed in Lecture 18
• Avoids computation of expansions that are not needed
• For practical problems was shown to achieve 10 times increase in speed.
• Are there other types of adaptivity possible?
Effect of using same $P$ throughout

- S|R translation to nearest neighborhood will probably control value of $P$
- This means that we are wasting precision on other translations.
  - For the Helmholtz equation $P$ grows with translation distance
- Idea evaluate initial expansions only till needed error …
  - Discussed this for Helmholtz Equation
  - Others have implemented it for Laplace
- So, algorithm must be carefully analyzed, and a translation length dependent strategy figured out …
Computation of the Translation Operator

- Consider we evaluate the translation via matrix multiplication
- The translation matrix elements have to be computed
- $O(P^2)$ elements imply a complexity of at least $O(P^2)$
- In practice it can be much higher
  - Using Clebsch-Gordan coefficients evaluating matrix coefficients for Helmholtz requires $O(P^{5/2})$
  - Recursive evaluation, as discussed for Laplace and Helmholtz equation in the previous class, can reduce complexity to $O(P^2)$
  - Exploitation of Symmetries gets us to $O(P^{3/2})$
  - Exploitation of diagonal structure in transform space gets us to $O(P \log P)$
Efficient Exact Translation

• Sometimes the translation can be decomposed into simpler translations along coordinate directions

• Correspond to a tensor product approximation or (Kronecker product approximation) or “separation of variables”

• Plausibility \[ C = A \otimes B \text{ or } C = A \otimes I + I \otimes B \]

• Using KP matrix vector product needs just \( O(N^{3/2}) \)

• Also an interesting research area
  – Interesting references thesis of Pitsianis, thesis of Florence, (working with C. van Loan at Cornell)
  – Paper of Sun and Pitsianis (SIREV, 2001)
Properties of translation matrix

- Additions of Translations cause composition of operators
  \[ [E|F](t_1 + t_2) = [E|F](t_1)[E|F](t_2) \]
- So now we can compute translation matrices of a “few” basis directions in the hierarchy
- Other operators can be obtained as compositions
- But matrix multiplication is \( O(P^3) \) …
- However can try to seek low-rank sums of decomposition of the matrix
Efficient approximate translation

- Translation matrices may have most of their “energy” in a few subspace dimensions.

Approximate SVD factorization

\[ [E|F] = U\Sigma V^* = \sum_{j=1}^{p} \sigma_j u_j v_j^* \approx \sum_{j=1}^{r} \sigma_j u_j v_j^* \]

Then the matrix vector product

\[ [E|F] \{ \alpha \} = \{ \beta \}, \quad \beta_i = [E|F]_{ij} \alpha_j \]

can be approximated as

\[ [E|F] \{ \alpha \} = [U\Sigma V^T] \{ \alpha \} \approx \sum_{n=1}^{r} \sigma_j u_j (v_j^*, \alpha) \]
Review of the course
Factorization: A very simple algorithm

- Not FMM, but has some key ideas
- Consider

\[ S(x_i) = \sum_{j=1}^{N} \alpha_j (x_i - y_j)^2 \quad i = 1, \ldots, M \]

- 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
Far Field and Near Field

Near Field of the $i$th source:

\[ |y - x_i| < r_c. \]

Far Field of the $i$th source:

\[ |y - x_i| > R_c. \]

What are these $r_c$ and $R_c$?
depends on the potential + some conventions for the terminology
Local (Regular) Expansion

Do not confuse with the Near Field!

Let

We call expansion local (regular) inside a sphere if the series converges for $\forall y$, $|y - x_*| < r_*$. 

We also call this R-expansion, since basis functions $R_m$ should be regular.
Local Expansion of a Regular Potential

Can be like this:

\[ |y - x^*| < r^* < |x_i - x^*| \]

\[ r^* > |y - x^*| > |x_i - x^*| \]

...or like this:

\[ |y - x^*| < r^* < |x_i - x^*| \]

\[ r^* > |y - x^*| > |x_i - x^*| \]
Local Expansion of a Singular Potential

Can be like this:

- $|y - x_*| < r_* \leq |x_i - x_*|$

Like this only!

...or like this:

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

Never ever!

Because $x_i$ is a singular point!
Use Compression!

Compression operator:

\[ A^n = \text{Compress}(a^n) \]

Required Property:

\[ a^n \cdot b^n = \text{Compress}(a^n) \cdot \text{Compress}(b^n). \]

Consider \( \mathbb{R}^2 \):

\[ a^n \cdot b^n = (a \cdot b)^n = (a_1b_1 + a_2b_2)^n \]

\[ = a_1^n b_1^n + \binom{n}{1} a_1^{n-1} b_1^{n-1} a_2 b_2 + \binom{n}{2} a_1^{n-2} b_1^{n-2} a_2^2 b_2^2 + \ldots + a_2^n b_2^n \]

The length is only \((n + 1)\), not \(2^n\)

Let us define:

\[ A^n = \text{Compress}(a^n) = \begin{pmatrix} a_1^n \sqrt{\binom{n}{1}} a_1^{n-1} a_2 \sqrt{\binom{n}{2}} a_1^{n-2} a_2^2 \ldots a_2^n \end{pmatrix}, \]

\[ B^n = \text{Compress}(b^n) = \begin{pmatrix} b_1^n \sqrt{\binom{n}{1}} b_1^{n-1} b_2 \sqrt{\binom{n}{2}} b_1^{n-2} b_2^2 \ldots b_2^n \end{pmatrix} \]
Example of Fast Computation

\[ v_j = \sum_{i=1}^{N} u_i \Phi(y_j, x_i) = \sum_{m=0}^{p-1} c_m \cdot \left( y_j - x_\ast \right)^m + \text{Residual}, \quad c_m = \frac{1}{m!} \sum_{i=1}^{N} u_i e^{x_i^\ast \cdot x_i} x_i^m. \]

Equivalent to:

\[ v_j = \sum_{m=0}^{p-1} C_m \cdot \text{Compress}\left( \left( y_j - x_\ast \right)^m \right) + \text{Residual}, \quad C_m = \frac{1}{m!} \sum_{i=1}^{N} u_i e^{x_i^\ast \cdot x_i} \text{Compress}(x_i^m). \]

Number of multiplications (complexity) to obtain \( v_j \):

\[ \text{Complexity} = 1 + 2 + \ldots + p = \frac{p(p + 1)}{2}. \]
Compression Can be Performed for any Dimensionality (Example for 3D):

\[ a^n \cdot b^n = (a \cdot b)^n = (a_1b_1 + a_2b_2 + a_3b_3)^n \]

\[ = [(a_1b_1 + a_2b_2) + a_3b_3]^n = \sum_{m=0}^{n} \binom{n}{m} (a_1b_1 + a_2b_2)^{n-m} a_3^m b_3^m \]

\[ = \sum_{m=0}^{n} \sum_{l=0}^{n-m} \binom{n}{m} \binom{n-m}{l} a_1^{n-m-l} b_1^{n-m-l} a_2^l b_2^l a_3^m b_3^m \]

\[ = a_1^n b_1^n + \binom{n}{1} a_1^{n-1} b_1^{n-1} a_2 b_2 + \binom{n}{2} a_1^{n-2} b_1^{n-2} a_2^2 b_2^2 + \ldots + a_2^n b_2^n \]

\[ + \binom{n}{1} a_1^{n-1} b_1^{n-1} a_3 b_3 + \binom{n}{1} \binom{n-1}{1} a_1^{n-2} b_1^{n-2} a_2 b_2 a_3 b_3 + \ldots + a_3^n b_3^n \]

\[ \text{Compress}(a^n) = \left( a_1^n, \binom{n}{1} a_1^{n-1} a_2, \binom{n}{2} a_1^{n-2} a_2^2, \ldots, a_2^n \right) \]

The length of \( a^n \) is \( (n+1)+n+\ldots+1 = (n+1)(n+2)/2 \)
Four Key Stones of FMM

• Factorization
• Error
• Translation
• Grouping
Summary of formal requirements for functions that can be used in FMM

- We have two sets of points:
  \[ X = \{ x_1, x_2, \ldots, x_N \}, \quad x_i \in \mathbb{R}^d, \quad i = 1, \ldots, N, \]
  \[ Y = \{ y_1, y_2, \ldots, y_M \}, \quad y_j \in \mathbb{R}^d, \quad j = 1, \ldots, M. \]

- We have functions (potentials):
  \[ \Phi(x_i, y) : \mathbb{R}^d \to \mathbb{R}, \quad y \in \mathbb{R}^d, \quad i = 1, \ldots, N. \]

- These functions can be factorized as (local expansion):
  \[ \Phi(x_i, y) = A(x_i, x_*) \circ R(y - x_*), \quad |y - x_*| < r < |x_i - x_*|, \quad i = 1, \ldots, N \]

- These functions can be factorized as (far field expansion):
  \[ \Phi(x_i, y) = B(x_i, x_*) \circ S(x - x_*), \quad |y - x_*| > R > |x_i - x_*|, \quad i = 1, \ldots, N \]

- The product is distributive operation with respect to addition
  \[ (u_1A_1 + u_2A_2) \circ F = u_1A_1 \circ F + u_2A_2 \circ F, \quad F = S, R \]
Summary of formal requirements for functions that can be used in FMM (2)

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

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

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

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

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

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

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

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

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

- And we are looking for sums:

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

- Some generalization are possible, say instead of $\Phi(\mathbf{y}_j, \mathbf{x}_i)$ we can consider $\Phi_i(\mathbf{y}_j)$, etc.
Middleman Algorithm

Standard algorithm

Sources

Evaluation Points

Middleman algorithm

Sources

Evaluation Points

Total number of operations: $O(NM)$

Total number of operations: $O(N+M)$
Idea of a Single Level FMM

**Standard algorithm**

- Sources
- Evaluation Points

\[ \text{Total number of operations: } O(NM) \]

**SLFMM**

- Sources
- Evaluation Points
- \( L \) groups
- \( K \) groups

\[ \text{Total number of operations: } O(N+M+KL) \]
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
Hierarchical Spatial Domains

$E_1$  

$E_2$  

$E_3$  

$E_4$
UPWARD PASS

- Partition sources into a source hierarchy.
- Stop hierarchy so that boxes at the finest level contain $s$ 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 $\Phi(x_i,y) = \sum u_i B(x^*,x_i) S(x^*,y)$.

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

$$C^{(n,L)} = \sum_{x_i \in B_1(n,L)} u_i B(x_i, x^{(n,L)}_c).$$

- We need to keep these coefficients. $C^{(n,l)}$ 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?)
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 $n=l,...,2^ld\quad l=L,...,2$
• 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
• $S$ expansion is valid in the domain $E_3$ outside domain $E_1$ (provided $d<9$)
DOWNWARD PASS

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

$$\Phi_{4}^{(n,l)}(y) = \tilde{D}^{(n,l)} \circ R(y - x_{c}^{(n,l)})$$

$$\tilde{D}^{(n,l)} = \sum_{m \in \mathcal{I}_{4}(n,l)} (S|R)(x_{c}^{(n,l)} - x_{c}^{(m,l)})C(m,l).$$

- 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
Downward Pass. Step 1.

Level 2:

Level 3:
Downward Pass. Step 1.

THIS MIGHT BE
THE MOST EXPENSIVE
STEP OF THE ALGORITHM
Downward Pass. Step 1.

\[ P_4 = \text{PowerOfE_4Neighborhood} = 3^d2^d - 3^d = 3^d(2^d - 1) \]

- \( d = 1 \): \( P_4 = 3 \),
- \( d = 2 \): \( P_4 = 27 \),
- \( d = 3 \): \( P_4 = 189 \),
- \( d = 4 \): \( P_4 = 1215 \),

Exponential Growth

Total number of S|R-translations per 1 box in \( d \)-dimensional space (far from the domain boundaries)
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

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

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

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

$$
D^{(n,l)} = \tilde{D}^{(n,l)} + (R|R) \left( x_c^{(n,l)} - x_c^{(m,l-1)} \right) D^{(m,l-1)}, \quad m = \text{Parent}(n).
$$

\[ \text{.} \]

$$
\Phi_4^{(n,l)}(y) = \tilde{D}^{(n,l)} \circ R(y - x_c^{(n,l)}),
$$

$$
\tilde{D}^{(n,l)} = \sum_{m \in \text{I}_4(n,l)} (S|R) \left( x_c^{(n,l)} - x_c^{(m,l)} \right) C^{(m,l)}.
$$
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 $E_3(n, l + 1)$:

$$E_3(n, l + 1) = E_3(n, l) \cup E_4(n, l + 1).$$
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
2^d-trees

<table>
<thead>
<tr>
<th>Level</th>
<th>2-tree (binary)</th>
<th>2^2-tree (quad)</th>
<th>2^d-tree</th>
<th>Number of Boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Parent</td>
<td></td>
<td></td>
<td>2^d</td>
</tr>
<tr>
<td>2</td>
<td>Neighbor (Sibling)</td>
<td>Self</td>
<td>Maybe Neighbor</td>
<td>2^{2d}</td>
</tr>
<tr>
<td>3</td>
<td>Children</td>
<td></td>
<td></td>
<td>2^{3d}</td>
</tr>
</tbody>
</table>
Parent Number

Parent numbering string:

\[
    \text{Parent}(N_1, N_2, ..., N_{l-1}, N_l) = (N_1, N_2, ..., N_{l-1}).
\]

Parent number:

\[
    \text{Parent}(\text{Number}) = (2^d)^{l-2} \cdot N_1 + (2^d)^{l-3} \cdot N_2 + ... + N_{l-1}.
\]

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

\[
    \text{Parent}(11_{10}) = \text{Parent}(23_4) = 2_4 = 2_{10}.
\]

Parent’s universal number:

\[
    \text{Parent}((\text{Number}, l)) = (\text{Parent}(\text{Number}), l - 1).
\]

Algorithm to find the parent number:

\[
    \text{Parent}(\text{Number}) = \left\lfloor \frac{\text{Number}}{2^d} \right\rfloor
\]

For box #23_4 (gray or black) the parent box number is 2_4.
Children Numbers

Children numbering strings:

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

Children numbers:

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

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

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

Children universal numbers:

$$\text{Children}((\text{Number}, l)) = (\text{Children}(\text{Number}), l + 1).$$

Algorithm to find the children numbers:

$$\text{Children}(\text{Number}) = \{2^d \cdot \text{Number} + j\}, \quad j = 0, ..., 2^d - 1,$$
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. \( \lfloor 5981 / 8 \rfloor = 747 \).

**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.