On the Hierarchical Modeling Technique with Applications

Jingfang Huang
Department of Mathematics
UNC at Chapel Hill

Sample hierarchical models and algorithms

Fast algorithms

- multigrid
- FFT: fast Fourier Transform
- FMM: fast multipole method
- fast direct solvers and H-matrix

Recent models

- convolutional neural network
- multi-level models in statistics (hierarchical linear models or nested data models)
- ……

Basics of the Hierarchical algorithms and models

Note: Hierarchical algorithms use **divide-and-conquer**, **compression**, and **hierarchical tree** algorithm design paradigm.

1. Hierarchical Tree Structure

Data is processed on a hierarchical tree structure

Examples:

multigrid tree structure,

FFT odd/even based tree structure,

fast multipole quad- and octree-tree structures,

fast direct solvers and H-matrix

convolutional neural network (CNN)

multi-level models in statistics (hierarchical linear models or nested data models)

Uniform binary tree

Assume there are N leaf nodes (N pieces of information).

Then there will be log(N) levels,

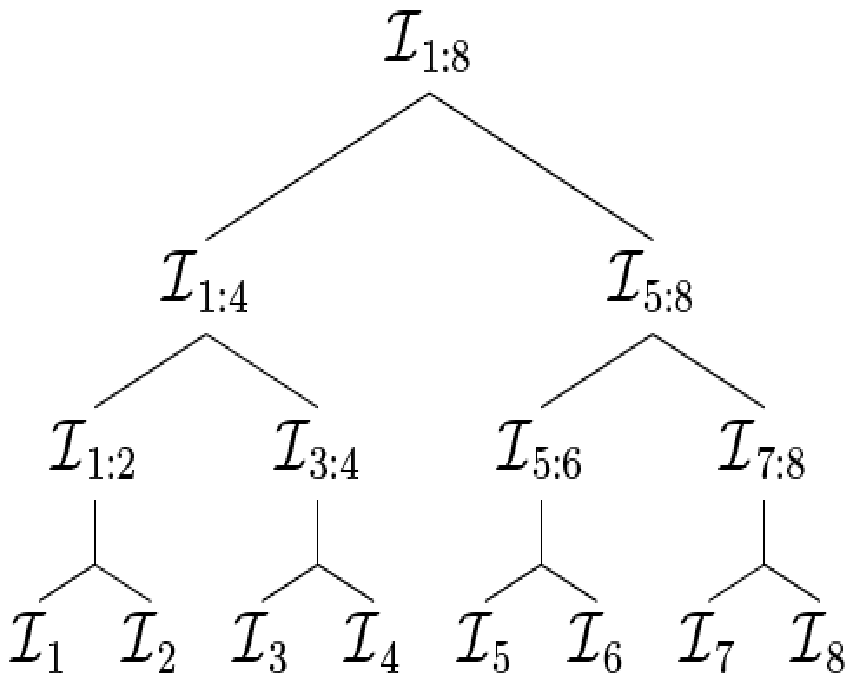
There will be O(N) nodes

If O(N) work at each level,

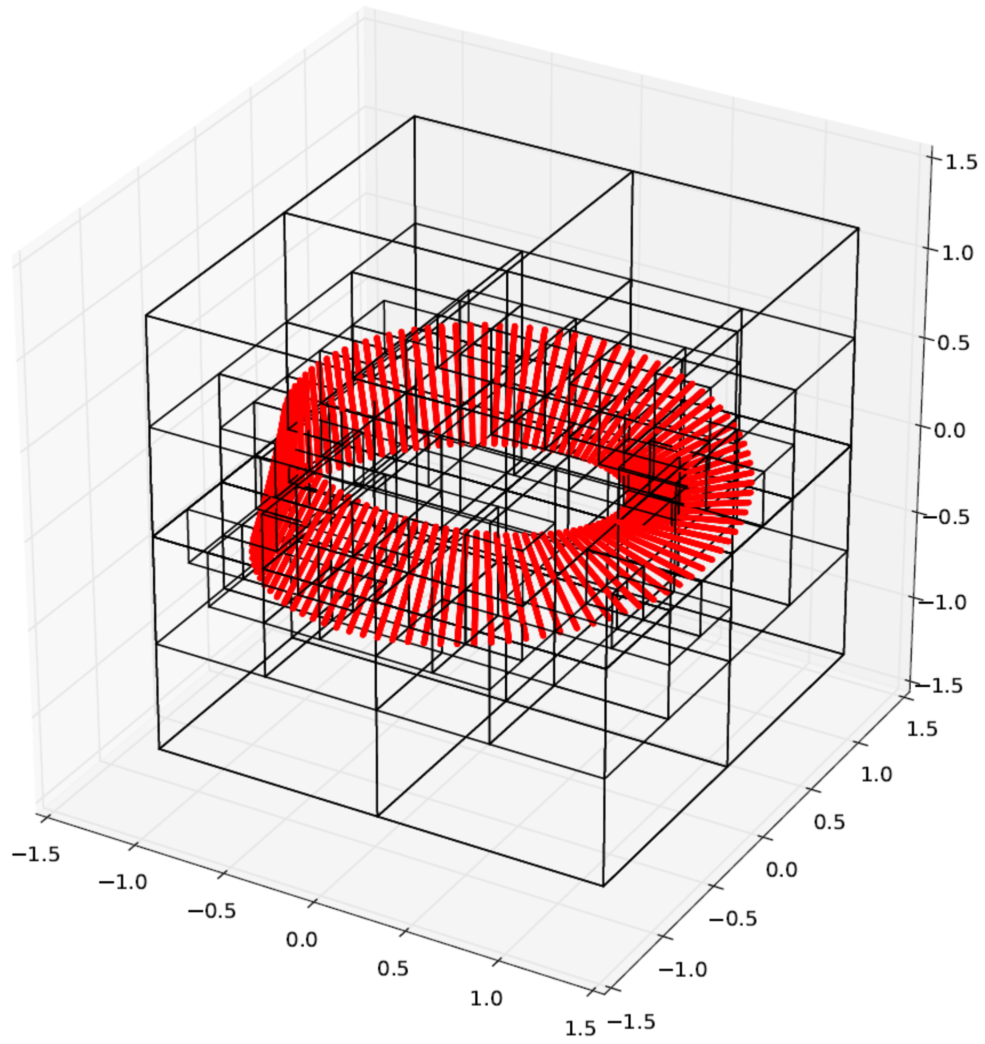
=>*O(N log(N))* **algorithm**

If O(1) work at each node,

=> *O(N)* **algorithm.**

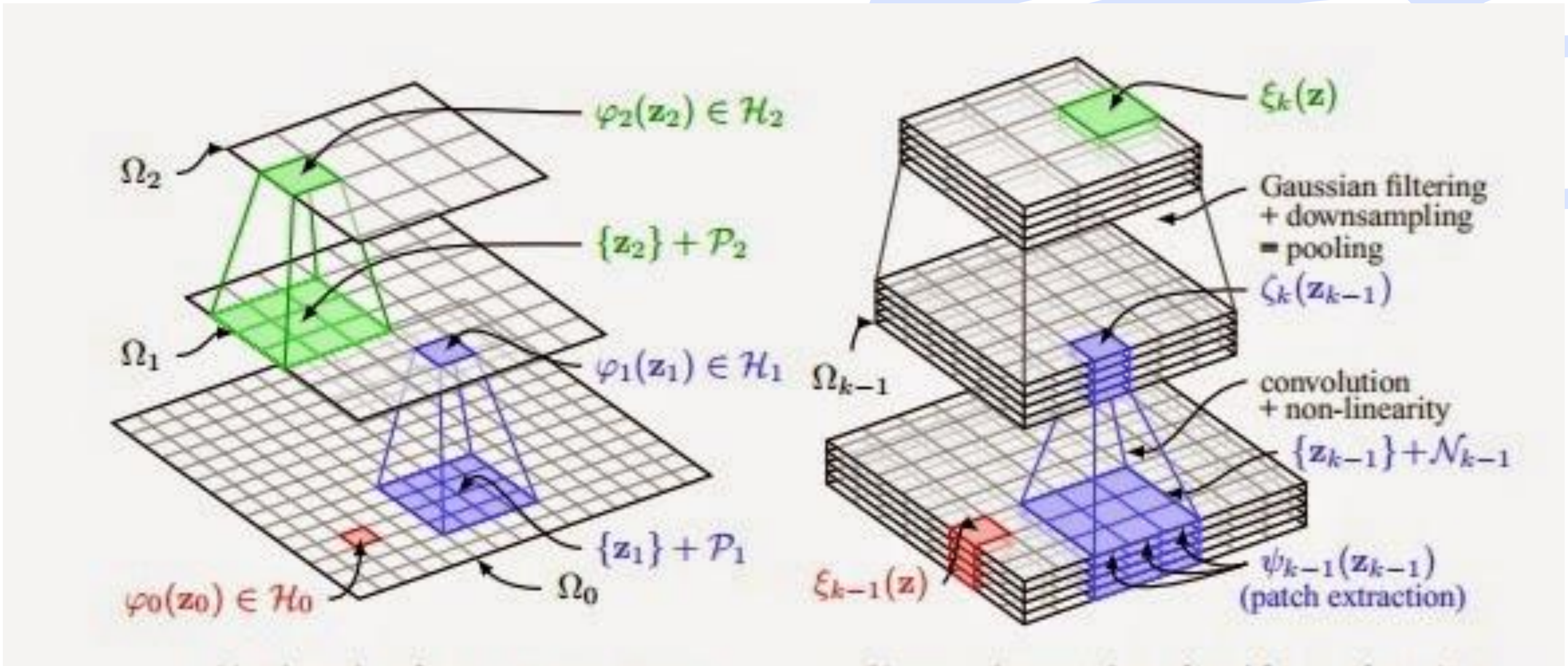


Spatial Adaptive FMM octree



From: Manas Rachh

Convolutional Neural Networks



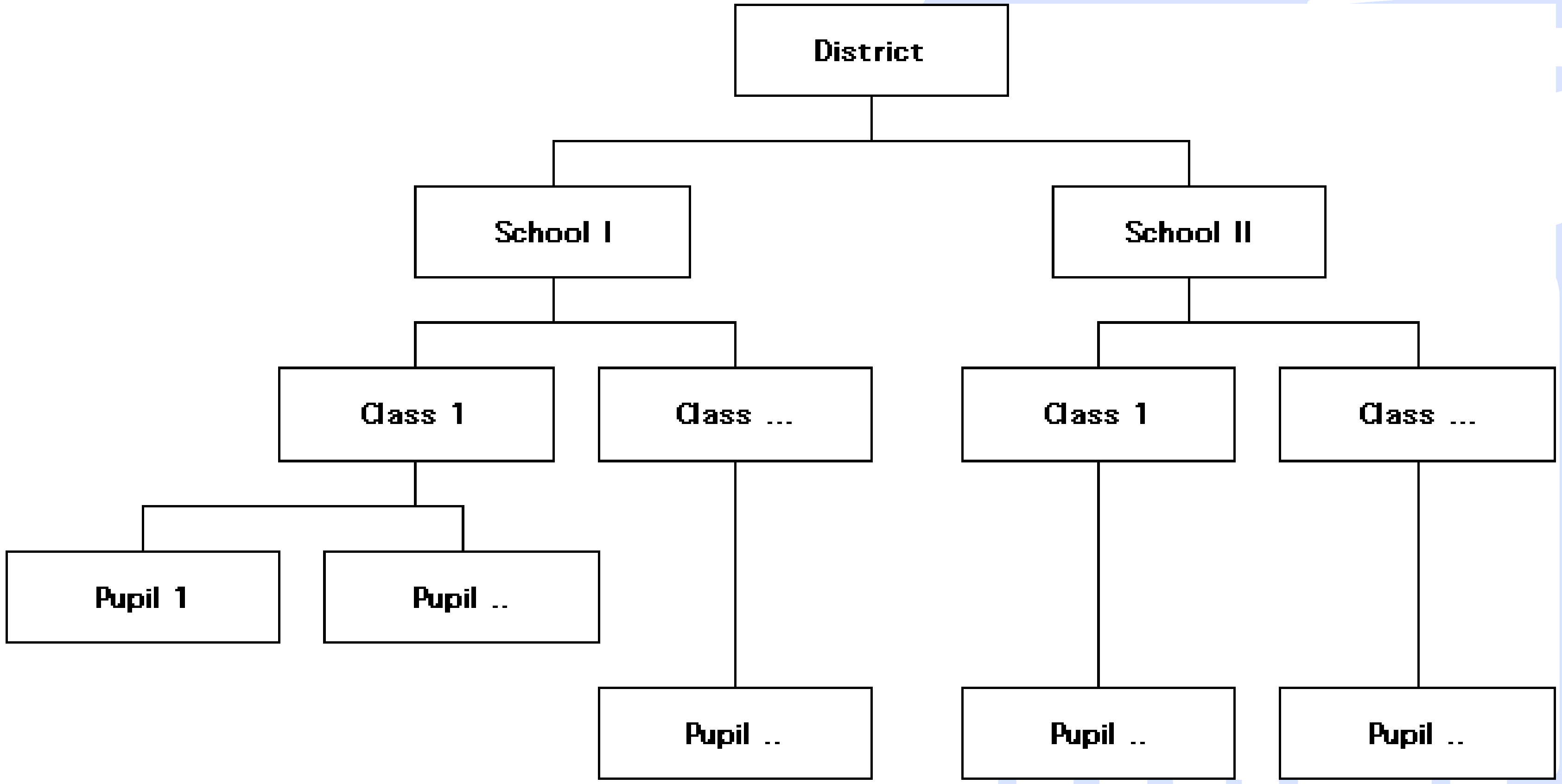
Ω_2
$\varphi_2(\mathbf{z}_2) \in \mathcal{H}_2$
$\{\mathbf{z}_2\} + \mathcal{P}_2$
Ω_1
$\varphi_1(\mathbf{z}_1) \in \mathcal{H}_1$
$\{\mathbf{z}_1\} + \mathcal{P}_1$
$\varphi_0(\mathbf{z}_0) \in \mathcal{H}_0$
Ω_0
$\xi_k(\mathbf{z})$
Gaussian filtering + downsampling = pooling
$\zeta_k(\mathbf{z}_{k-1})$
Ω_{k-1}
convolution + non-linearity
$\{\mathbf{z}_{k-1}\} + \mathcal{N}_{k-1}$
$\xi_{k-1}(\mathbf{z})$
$\psi_{k-1}(\mathbf{z}_{k-1})$ (patch extraction)

(a) Hierarchy of Image Feature Maps (b) Zoom between layer k and $k-1$ of our CKN

Figure 1: Left: concrete representation of the successive layers for the multilayer convolutional kernel. Right: one layer of the convolutional neural network that approximates the kernel.

https://raweb.inria.fr/rapportsactivite/RA2014/lear/uid45.html

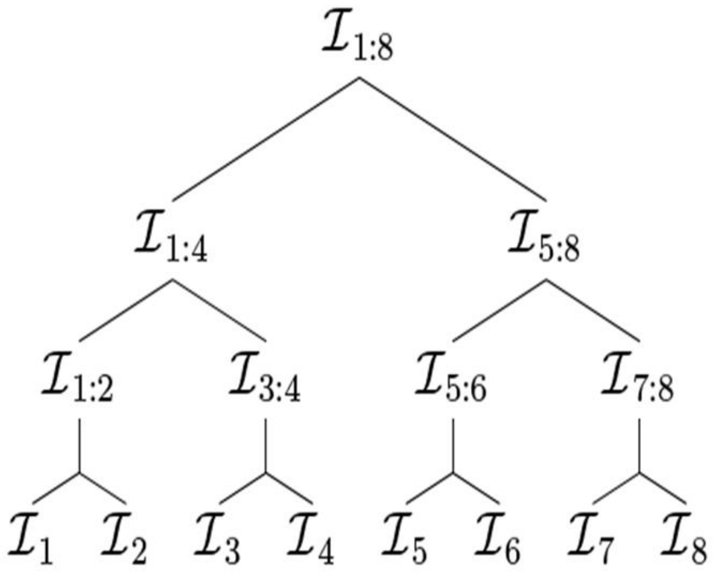
Hierarchical linear model in statistics



http://randomvariation.blogspot.com/2011/12/hierarchical-linear-models-overview.html

Related concepts

- Parent
- Children
- Neighbors
- 2nd nearest neighbors
- Interaction list (FMM)
- Far field
- Near field
- …



Questions: How about more general networks, e.g., social network? How to define distance? Distance between different websites or different stocks? How to generate the hierarchical tree for high dimensional data located on a low dimensional hypersurface? How to generate the tree structure in parallel?

2. Data compression

Data/information can be compressed!

Examples:

Multigrid: “low-frequency” information

FFT: halving lemma

FMM: mulitipole and local expansions

fast direct solvers/H-matrix: low-rank representations

convolutional neural network: low-dimensional models

Identify the low rank/low dimensional/compact structures in a system.

Separation of Variables

Low rank and low dimensional data

$$\underset{\substack{\downarrow\\ \text{2D}}}{f(x,y)} \cong \sum_{n=1}^{30} \underset{\text{1D}}{\underline{f_n(x)}}\;\underset{\text{1D}}{\underline{g_n(y)}}$$

Low rank vs. low dimensional.

Question: how to find low dimensional structure in high dimensional data set? Proper representations of low-dimensional data?

3. "Local" Translations on the tree

Translations are **"local"**, and are performed on the compressed data - a node only interacts with its parent, children, and maybe siblings (FMM interaction list)

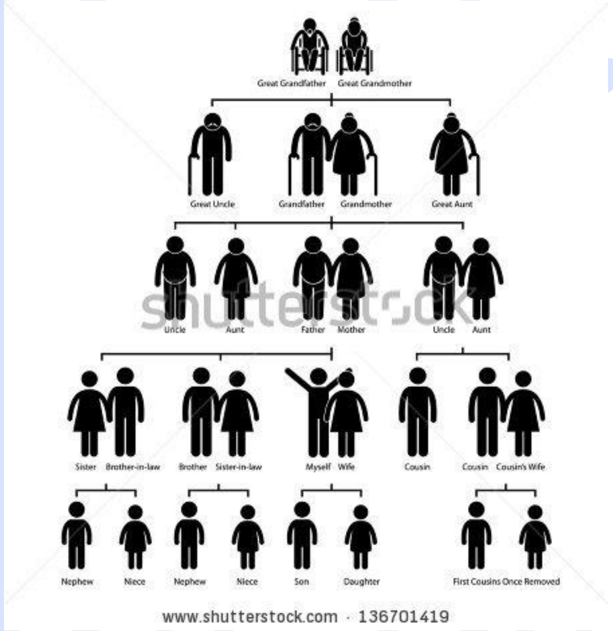
Examples:

FFT: T(N) = 2 T(N/2) + O(N)

FMM: M2M, M2L, L2L.

Fast Direct Solver: Schur Com., Woodbury Matrix Id.

Convolutional Neural Network: local connectivity and filter for compressed information.

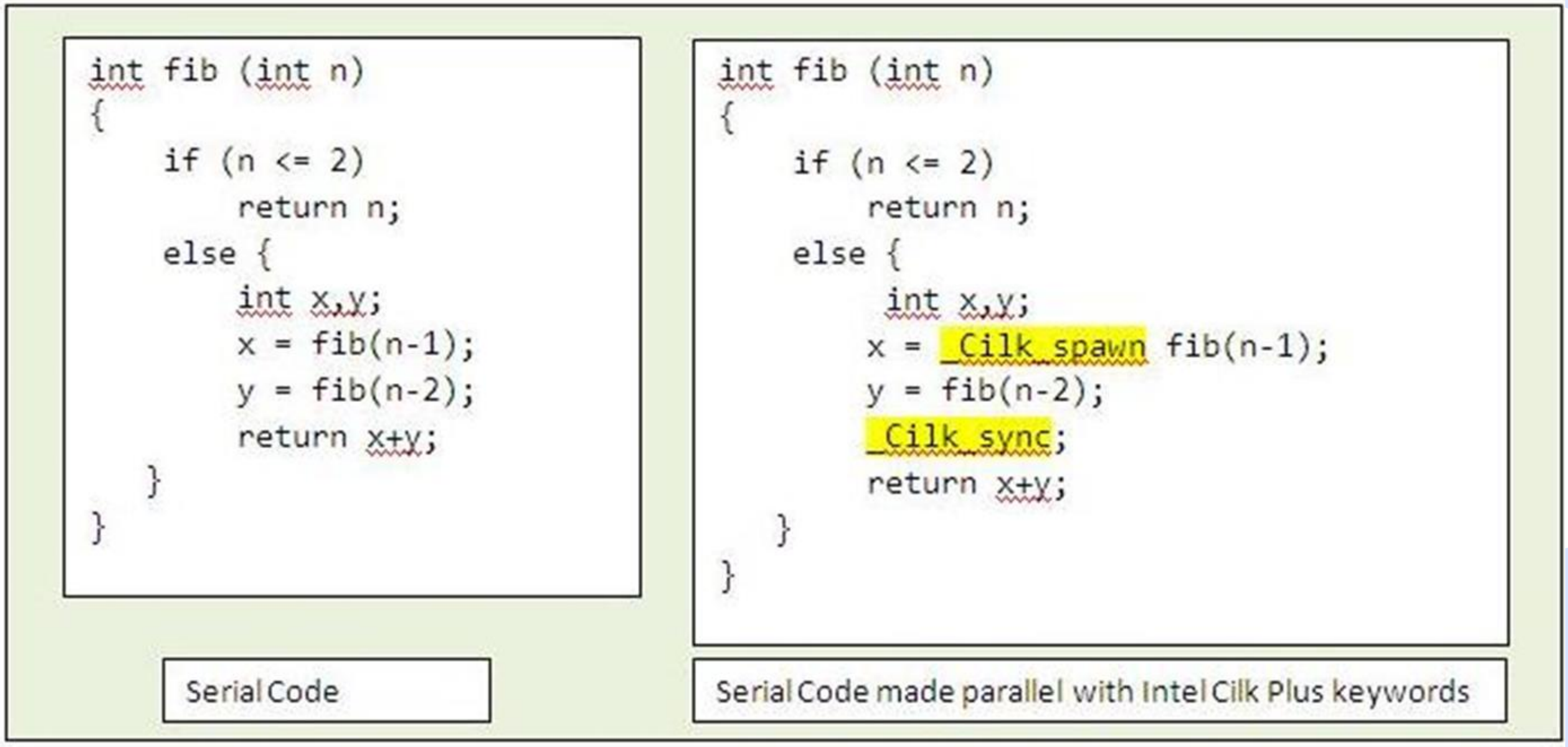


Note: current non-local models (e.g. fractional differential equation models) become "local" if considered on the tree.

Q: How to find the right "local" translation operators?

4. Recursive Thinking and Programming

Models and algorithms can be designed and implemented in a recursive fashion, which can be parallelized using dynamical schedulers (e.g., Cilk, HPX), or be flattened for improved parallel performance.



```
int fib (int n)
{
    if (n <= 2)
        return n;
    else {
        int x,y;
        x = fib(n-1);
        y = fib(n-2);
        return x+y;
    }
}
```

Serial Code

```
int fib (int n)
{
    if (n <= 2)
        return n;
    else {
         int x,y;
        x = _Cilk_spawn fib(n-1);
        y = fib(n-2);
        _Cilk_sync;
        return x+y;
    }
}
```

Serial Code made parallel with Intel Cilk Plus keywords

Hierarchical Modeling Technique

This technique first identifies any low-rank, or low-dimensional, or other compact features using appropriate and mathematically rigorous definitions. The compressed representations are then recursively collected from children to parents, and transmitted "locally" between different nodes on a hierarchical tree structure.

In model design and numerical implementation, the hierarchical models can be expressed as recursive algorithms, which can be interfaced with existing dynamical scheduling tools or flattened using techniques from the HPC community for improved parallel efficiency.

Some Examples

- FFT
- FMM
- Multigrid
- …

- How about other problems?

Hierarchical algorithms and energy minimization???

Boundary value elliptic PDEs:

u''(x)+p(x) u'(x) + q(x) u(x)=f(x)
Boundary conditions

$\Leftrightarrow$

There exists an excellent hierarchical algorithm for this problem (Greengard/Lee) ✓

Energetic Variation Formulation

Minimize E(u)
Boundary Conditions

Most optimization methods are iterative?

?

Can we use the principles in the hierarchical models/algorithms to more efficiently solve the optimization problems???

Interactions(potential theory)? Energy(variational formulation)? And PDEs?

"The basic mechanisms for many PDEs are different interactions between particles (e.g., the Coulomb interactions in Poisson equations, the equations of states in Navier-Stokes equations, and the assumptions for different diffusions). The nature of the interactions often gives rise to (nonlocal) integral form models. Many pure PDE forms (e.g., diffusion and transport equations) can be viewed as approximation/truncations of the nonlocal interactions, and are the results of averaging and limiting." – Professor Chun Liu

And we know the interactions are often "compressible".

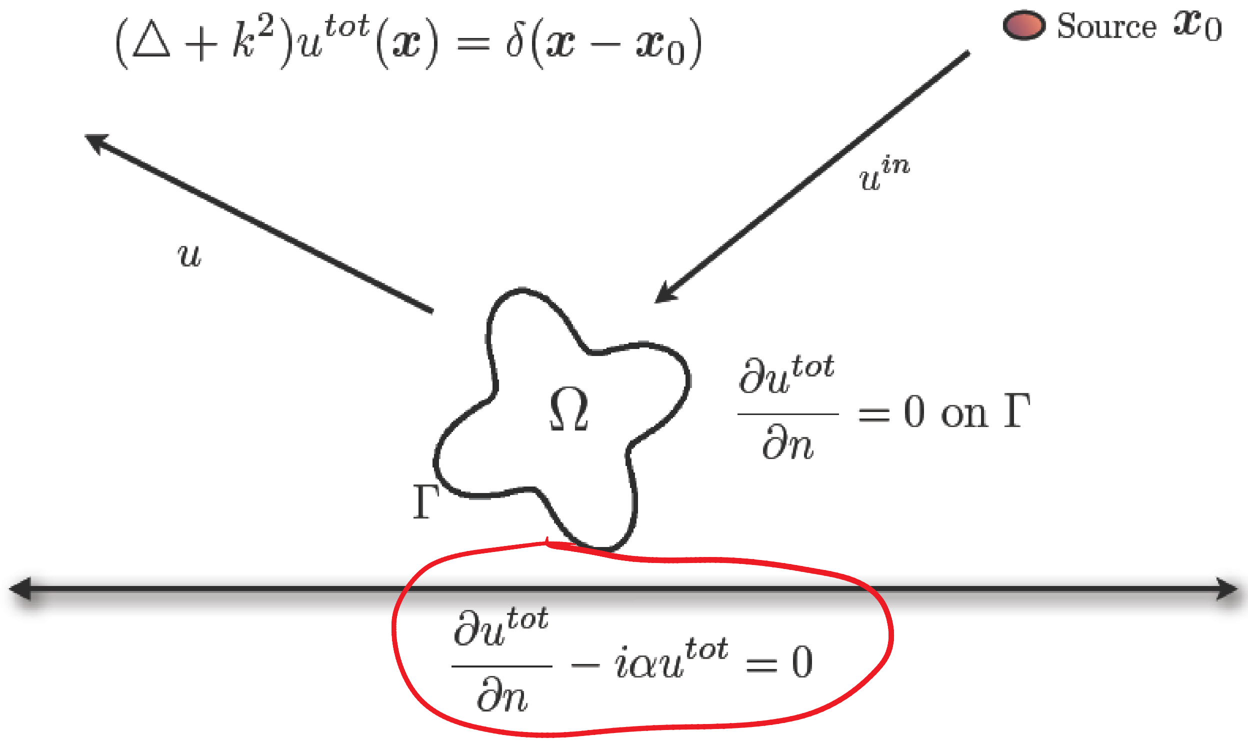
Case Study 01

Waves in layered-media

Joint work with Min Hyung Cho (UMass Lowell), Dangxing Chen (UC Berkeley), and Wei Cai (SMU)

Background: Layered media Green's function

A sample two layered media



$(\Delta + k^2)u^{tot}(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x}_0)$
Source $\boldsymbol{x}_0$
u^{in}
u
Ω
Γ
$\frac{\partial u^{tot}}{\partial n} = 0$ on Γ
$\frac{\partial u^{tot}}{\partial n} - i\alpha u^{tot} = 0$

Figure 1: Scattering from a *sound-hard* obstacle above an impedance plane.

"On the efficient representation of the half-space impedance Green's function for the Helmholtz equation", by Michael O'Neil, Leslie Greengard, Andras Pataki

Background: Integral equation methods

- **Integral equation formulation**

$$u^{tot}(x) = \int_\Gamma g_{k,\alpha}(x,y)\,\sigma(y)\,dy(s) + u^{in}(x),$$

where $g_{k,\alpha}(x,x_0)$ is the **domain Green's function** for the half-space with homogeneous impedance boundary conditions, and the resulting Fredholm 2nd kind integral equation become

$$-\frac{1}{2}\sigma(\boldsymbol{x}) + \int_\Gamma \frac{\partial}{\partial n_x} g_{k,\alpha}(\boldsymbol{x},\boldsymbol{y})\,\sigma(\boldsymbol{y})\,d\boldsymbol{y}(s) = -\frac{\partial}{\partial n_x} g_{k,\alpha}(\boldsymbol{x},\boldsymbol{x}_0)$$

Numerical difficulty: evaluating the layer potentials.

Domain Green's function: Method of Images and compression?

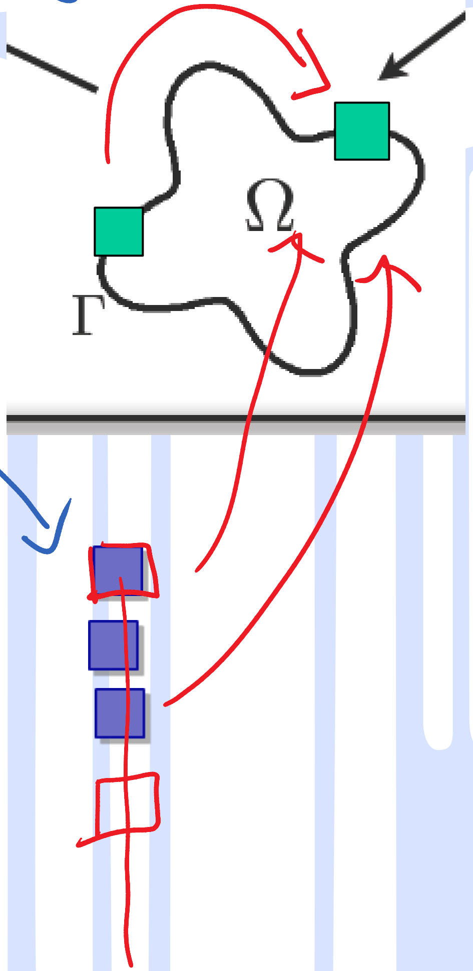
- **The method of images**

$\longrightarrow$ point image

$$g_{k,\alpha}(\boldsymbol{x},\boldsymbol{x}_0) = g_k(\boldsymbol{x},\boldsymbol{x}_0 - 2y_0\hat{\boldsymbol{y}})$$

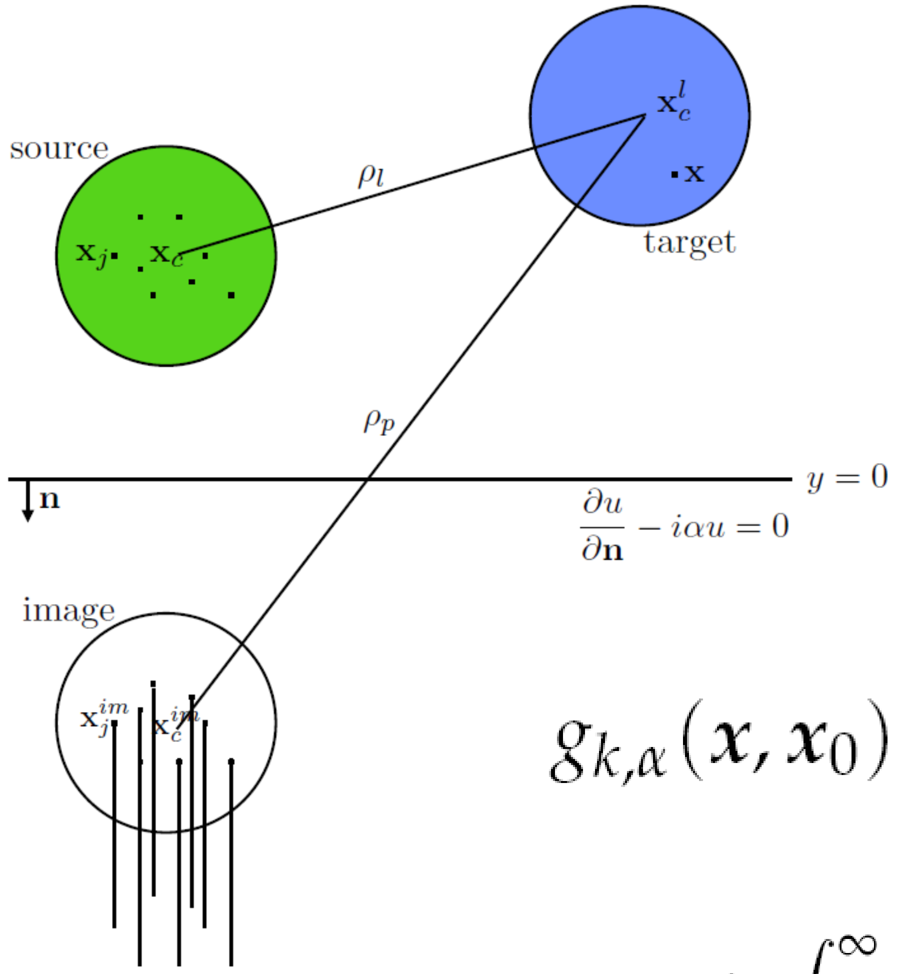
$$+\, 2i\alpha \int_0^\infty g_k(\boldsymbol{x},\boldsymbol{x}_0 - (2y_0+\eta)\hat{\boldsymbol{y}})\, e^{i\alpha\eta}\, d\eta,$$

line image



- How to compress? Transmit information?

Compression and translation



Info. compression of
(1) Original sources
(2) Point images
(3) Line images

$$g_{k,\alpha}(\boldsymbol{x}, \boldsymbol{x}_0) = g_k(\boldsymbol{x}, \boldsymbol{x}_0 - 2y_0\hat{\boldsymbol{y}})$$

$$+ 2i\alpha \int_0^\infty g_k(\boldsymbol{x}, \boldsymbol{x}_0 - (2y_0 + \eta)\hat{\boldsymbol{y}})\, e^{i\alpha\eta}\, d\eta,$$

Modified fast multipole method

- **Multipole-to-multipole:** as if the images do NOT exist.
- **Multipole-to-local:** modified translation operator (matrix) which include all the image contributions (spatially variant)
- **Local-to-local:** No change!

Error analysis: Note that source images are always well-separated from the target box.

M2L: analytical formula is possible. Either precomputed, or computed on the fly.

Multiple Layers: Sommerfeld Integral Representation

$$u^s(\mathbf{x}) = \int_{-\infty}^{\infty} \underbrace{\frac{e^{-\sqrt{\lambda^2-k^2}(y+y_c)}}{\sqrt{\lambda^2-k^2}} e^{i\lambda(x-x_c)}}_{\text{free-space info}} \underbrace{\left(\frac{1}{4\pi}\sum_{j=1}^{N} q_j e^{-\sqrt{\lambda^2-k^2}(y_j-y_c)} e^{i\lambda(x_c-x_j)}\right)}_{\text{uncompressed}} \underbrace{\left(\frac{\sqrt{\lambda^2-k^2}+i\alpha}{\sqrt{\lambda^2-k^2}-i\alpha}\right)}_{\text{image info}} d\lambda.$$

$$u^s(\mathbf{x}) \approx \int_{-\infty}^{\infty} \underbrace{\frac{e^{-\sqrt{\lambda^2-k^2}(y+y_c)}}{\sqrt{\lambda^2-k^2}} e^{i\lambda(x-x_c)}}_{\text{free-space info}} \underbrace{\left(\frac{1}{4\pi}\sum_{p=-P}^{P} \bar{\alpha}_p(-i)^p \left(\frac{\lambda-\sqrt{\lambda^2-k^2}}{k}\right)^p\right)}_{\text{compressed}} \underbrace{\left(\frac{\sqrt{\lambda^2-k^2}+i\alpha}{\sqrt{\lambda^2-k^2}-i\alpha}\right)}_{\text{image info}} d\lambda.$$

Yes, it is the fast multipole method.

For multiple layered media,

Form_MP and M2M (Compress Data): Only compress the free space kernel, no need to work on the matrix entries directly (domain Green’s function)

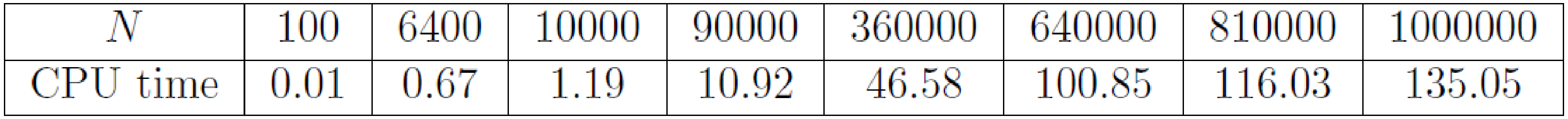
M2L (spatially variant translation): can be done using the Sommerfeld integral representation directly, includes all the “image contributions”

L2L: Same as free space FMM.

Numerical Results

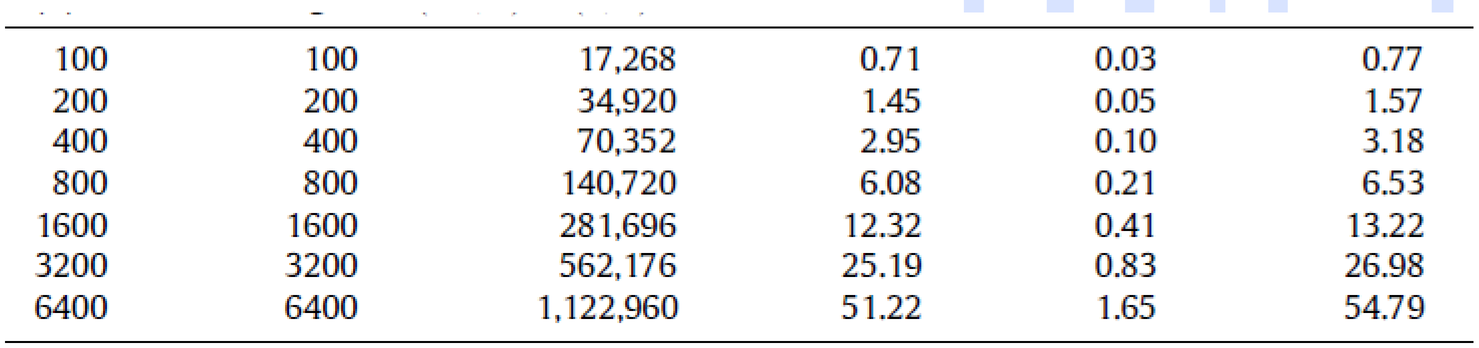
Avoid introducing many many images in the computation.

CPU time (seconds) for different N using $p = 39$ and $k = 0.1$



N	100	6400	10000	90000	360000	640000	810000	1000000
CPU time	0.01	0.67	1.19	10.92	46.58	100.85	116.03	135.05

As a comparison (from O'Neil, Greengard, and Pataki)



100	100	17,268	0.71	0.03	0.77
200	200	34,920	1.45	0.05	1.57
400	400	70,352	2.95	0.10	3.18
800	800	140,720	6.08	0.21	6.53
1600	1600	281,696	12.32	0.41	13.22
3200	3200	562,176	25.19	0.83	26.98
6400	6400	1,122,960	51.22	1.65	54.79

Current work

Multiple Layer and 3D codes…

Rigorous error analysis purely based on Sommerfeld integral decomposition.

………..

Case Study 2

Recursive Tree Algorithms for Orthogonal Matrix Generation and Matrix-Vector Multiplications

in

Rigid Body Brownian Dynamics Simulations

Joint work with Fuhui Fang (UNC), Gary Huber (UCSD), J. Andrew McCammon (UCSD), and Bo Zhang (Indiana)

Background: Rigid body dynamics

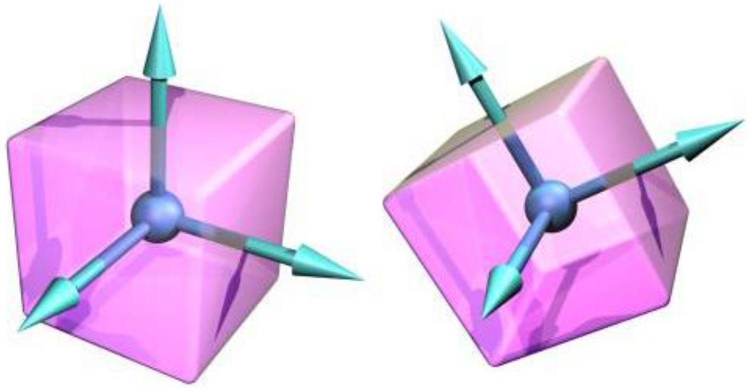
Consider a molecular system modeled by m rigid bodies, with a total of n “beads”

Given the force on and location of each bead

- $\vec{f_j} = (f_j^1, f_j^2, f_j^3)^T$ - External force
- $\vec{r_j} = (x_j, y_j, z_j)^T$ - Location

The **resultant force** (size 3) and **torque** (size 3)of the **rigid body** are given by

$$\vec{f_P} = \sum_{j=1}^{n} \vec{f_j}, \quad \vec{\tau}_P = \sum_{j=1}^{n} \vec{r_j} \times \vec{f_j},$$



Ermak-McCammon model: hydrodynamics interactions are modeled by ***Df=v***, D: Rotne-Prager-Yamakawa tensor, f: force on the beads, v: the velocity of the beads.

With rigid body constraints=>

$$\underbrace{Z_{6m\times 3n}\,\underbrace{(D^{-1})_{3n\times 3n}\,\underbrace{(Z^T)_{3n\times 6m}\begin{bmatrix}\mathbf{V}_1\\ \boldsymbol{\omega}_1\\ \vdots\\ \mathbf{V}_m\\ \boldsymbol{\omega}_m\end{bmatrix}}_{\text{velocities on beads}}}_{\text{forces on beads}}}_{\text{forces on rigid bodies}} = \underbrace{\begin{bmatrix}\mathbf{F}_1\\ \boldsymbol{\tau}_1\\ \vdots\\ \mathbf{F}_m\\ \boldsymbol{\tau}_m\end{bmatrix}}_{\text{given external forces}}$$

V ?

F ✓

Avoid computing D^{-1}:

Schur Complement (Gauss elimination)

$$\begin{bmatrix} 0 & Z \\ Z^T & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -F \\ 0 \end{bmatrix}$$

Orthogonal Linear Algebra

$x = V$

$$\begin{bmatrix} Q \\ \tilde{Q} \end{bmatrix}_{3N\times 3N} D^{-1}_{3N\times 3N} \begin{bmatrix} Q^T & \tilde{Q}^T \end{bmatrix}_{3N\times 3N} \begin{bmatrix} V \\ 0 \end{bmatrix}_{3N\times 1} = \begin{bmatrix} F \\ \tilde{F} \end{bmatrix}_{3N\times 1},$$

$$\begin{bmatrix} V \\ 0 \end{bmatrix} = \begin{bmatrix} Q^T & \tilde{Q}^T \end{bmatrix} D \left[Q^T F + \tilde{Q}^T \tilde{F}) \right]$$

$$= \begin{bmatrix} QD(Q^T F + \tilde{Q}^T \tilde{F}) \\ \tilde{Q}D(Q^T F + \tilde{Q}^T \tilde{F}) \end{bmatrix}.$$

Comparison of these formulations?

Problem Statement

Given

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_6 \end{bmatrix} = \begin{bmatrix} I & I & \dots & I \\ A_1 & A_2 & \dots & A_N \end{bmatrix},$$

Question 1: (BD with Hydrodynamics interactions)

Given the $6 \times (3N)$ matrix Z (or equivalently, $Q = \begin{bmatrix} q_1 \\ \cdots \\ q_6 \end{bmatrix}$), what is a good strategy to find the "remaining" $3N - 6$ orthonormal vectors $\tilde{Q} = \begin{bmatrix} q_7 \\ \cdots \\ q_{3N} \end{bmatrix}$, to form an orthogonal matrix $\begin{bmatrix} Q \\ \tilde{Q} \end{bmatrix}$?

Related Problems

- **Question 2: How to efficiently calculate**

$$\begin{bmatrix} Q \\ \tilde{Q} \end{bmatrix} \cdot \begin{bmatrix} \vec{F}_1 \\ \vec{F}_2 \\ \cdots \\ \vec{F}_N \end{bmatrix} ?$$

- **Question 3:**

$$\begin{bmatrix} Q^T & \tilde{Q}^T \end{bmatrix} \cdot \vec{V} ?$$

Note: this is an alternative to Schur complement with better stability for solving a constrained linear system from Brownian dynamics simulations.

Recursive thinking: divide and conquer

Parent-children relation on the tree structure

Parent

$$Z = \begin{bmatrix} H_1 & H_2 \end{bmatrix},$$

Two children:

child B child C

$$H_1 = \begin{bmatrix} I & I & \dots & I \\ B_1 & B_2 & \dots & B_M \end{bmatrix}_{6\times 3M}$$

$$H_2 = \begin{bmatrix} I & I & \dots & I \\ C_1 & C_2 & \dots & C_P \end{bmatrix}_{6\times 3P}$$

Recursive thinking: Assume children's problems are solved

- **Child 1:** $[T_1]_{(3M-6)\times 3M}$
- **Child 2:** $[T_2]_{(3P-6)\times 3P}$.
- **How about parent's problem?**

Easy part:

$$\left[\begin{array}{cc} \underbrace{H_1 \qquad\qquad H_2}_{} \\ \hdashline T_1 & 0_{(3M-6)\times 3P} \\ 0_{(3P-6)\times 3M} & T_2 \end{array}\right]_{(3M+3P-6)\times(3M+3P)}$$

— Parent's Z

We will "recycle" T_1 and T_2 (compressed, sparse vector)

Q: How to find the remaining 6 orthogonal vectors???

Recursive Thinking!

Only need modified Gram-Schmidt (or Householder reflection, or Givens rotation) **on 12 vectors.**

$$\left[\begin{array}{cc} H_1 & H_2 \\ 0 & H_2 \\ T_1 & 0 \\ 0 & T_2 \end{array}\right] \begin{array}{l} \left.\begin{array}{l} \}\ 6 \\ \}\ 6 \end{array}\right\} \boxed{MGS} \\ \}\ 3M-6 \quad \checkmark \\ \}\ 3P-6 \quad \checkmark \end{array}$$

$$T(N)=2*T(N/2)+cN$$ (solve using difference equations)

Some hard details

Algorithm details:

- **FMM-type upward pass for**

$$\begin{bmatrix} Q \\ \tilde{Q} \end{bmatrix} \cdot \begin{bmatrix} \vec{F}_1 \\ \vec{F}_2 \\ \cdots \\ \vec{F}_N \end{bmatrix} ?$$

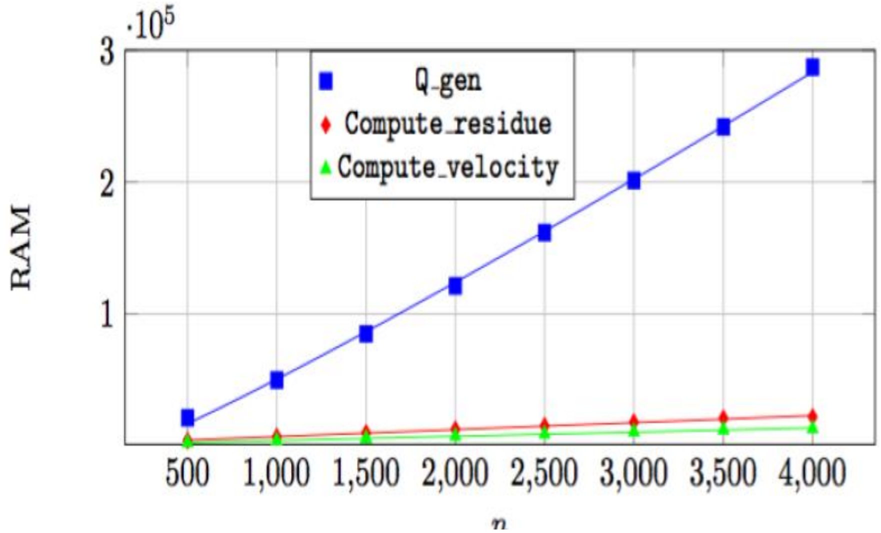
- **FMM-type downward pass for**

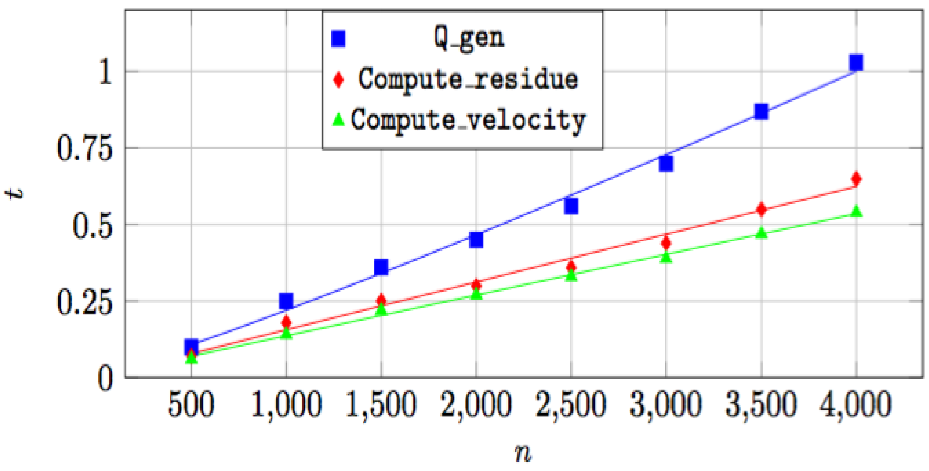
$$\begin{bmatrix} Q^T & \tilde{Q}^T \end{bmatrix} \cdot \vec{V} ?$$

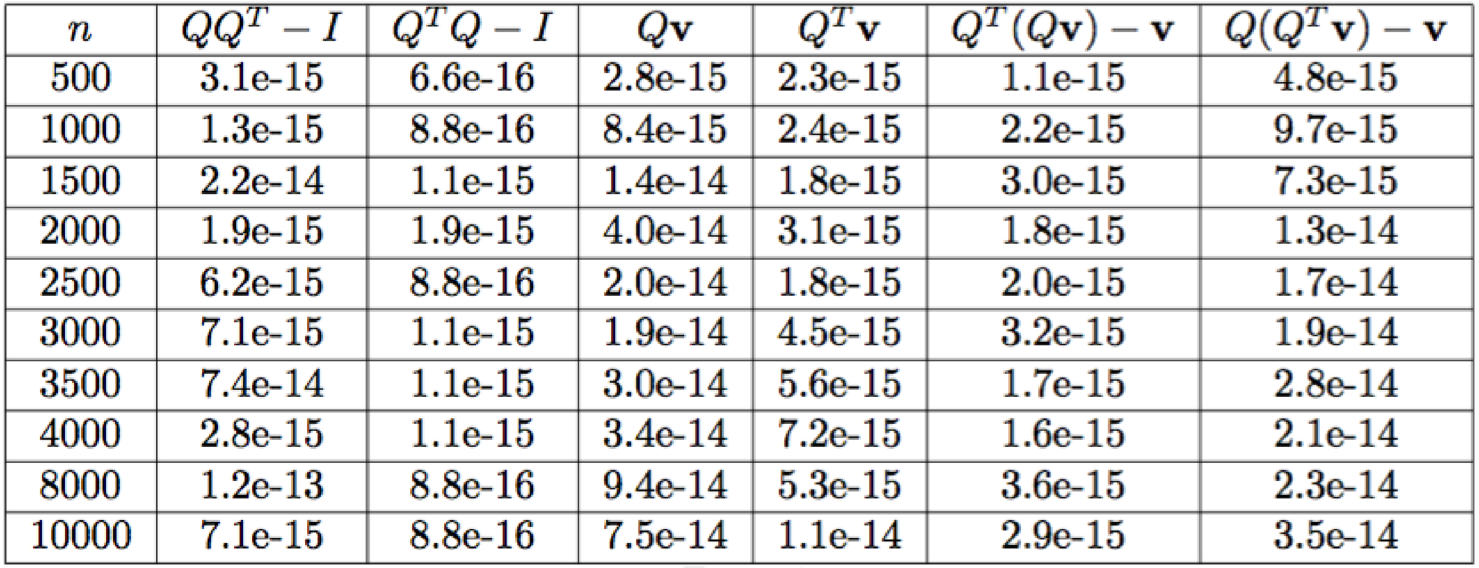
- **Need orthogonal linear algebra for better stability!**
- **Both passes require O(N) operations and O(N) storage** (with very small prefactors).

This was an undergraduate student's honors thesis project.

Numerical results







n	$QQ^T - I$	$Q^TQ - I$	$Q\mathbf{v}$	$Q^T\mathbf{v}$	$Q^T(Q\mathbf{v}) - \mathbf{v}$	$Q(Q^T\mathbf{v}) - \mathbf{v}$
500	3.1e-15	6.6e-16	2.8e-15	2.3e-15	1.1e-15	4.8e-15
1000	1.3e-15	8.8e-16	8.4e-15	2.4e-15	2.2e-15	9.7e-15
1500	2.2e-14	1.1e-15	1.4e-14	1.8e-15	3.0e-15	7.3e-15
2000	1.9e-15	1.9e-15	4.0e-14	3.1e-15	1.8e-15	1.3e-14
2500	6.2e-15	8.8e-16	2.0e-14	1.8e-15	2.0e-15	1.7e-14
3000	7.1e-15	1.1e-15	1.9e-14	4.5e-15	3.2e-15	1.9e-14
3500	7.4e-14	1.1e-15	3.0e-14	5.6e-15	1.7e-15	2.8e-14
4000	2.8e-15	1.1e-15	3.4e-14	7.2e-15	1.6e-15	2.1e-14
8000	1.2e-13	8.8e-16	9.4e-14	5.3e-15	3.6e-15	2.3e-14
10000	7.1e-15	8.8e-16	7.5e-14	1.1e-14	2.9e-15	3.5e-14

TABLE 1
Orthogonality preserving quality of Algorithms 1-3.

Take home message

A possible technique for developing fast algorithms for big data sets is to

- ❑ **Think recursively when design models/algorithms**
- ❑ **Find the special “compact” structures in the dataset**
- ❑ **Process the compressed information on a properly chosen hierarchical tree structure**
- ❑ **Translate Information “locally” on the tree structure**

Thanks!