Fast direct methods for structured matrices

Kenneth L. Ho (Stanford)

Math Colloquium, CMU, Oct. 2014

Introduction

Matrix problems are ubiquitous:

- $y = Ax$
- $x = A^{-1}b$
- $A = UV^*$
- $\Delta = \det A$

Introduction

Matrix problems are ubiquitous. However, they can be very expensive. For $A \in \mathbb{C}^{N \times N}$:

- $y = Ax$ — $O(N^2)$
- $x = A^{-1}b$ — $O(N^3)$
- $A = UV^*$ — $O(N^3)$
- $\Delta = \det A$ — $O(N^3)$

Classical methods are infeasible beyond $N \sim 10^4$.

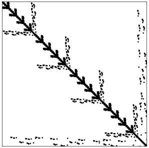
Introduction

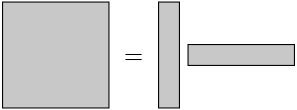
Matrix problems are ubiquitous. However, they can be very expensive. For $A \in \mathbb{C}^{N \times N}$:

- $y = Ax$ $\quad O(N^2)$
- $x = A^{-1}b$ $\quad O(N^3)$
- $A = UV^*$ $\quad O(N^3)$
- $\Delta = \det A$ $\quad O(N^3)$

Classical methods are infeasible beyond $N \sim 10^4$.

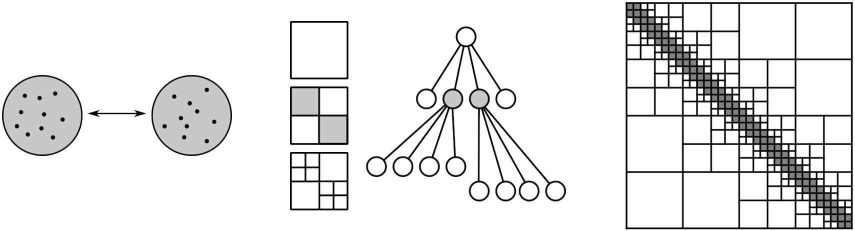
- Fortunately, many matrices in practice are **structured**
- Example: sparse or low-rank matrices
- Exploiting such structure can yield very efficient algorithms





Introduction

- **Hierarchical matrices**: low-rank submatrices at a hierarchy of scales
- Canonical example: N-body problem
  - Particle locations: x_i, $i = 1, \dots, N$
  - Interaction kernel: $K(x, y) = 1/\|x - y\|$
  - Forces: $f_i = \sum_{j=1}^{N} K(x_i, x_j)\, m_j$
- Matrix $A_{ij} = K(x_i, x_j)$ can be applied in $O(N)$ time using FMM [Greengard/Rokhlin]



- Applications in elliptic PDEs, integral equations, data analysis, etc.

Many hierarchical matrix problems can be solved efficiently using FMM.

- Example: $Ax = b$ using FMM + CG/GMRES
- Highly scalable, $O(n_{\text{iter}} N)$ complexity
- Very successful; industrial applications in electromagnetics, acoustics, etc.

Many hierarchical matrix problems can be solved efficiently using FMM.

- Example: $Ax = b$ using FMM + CG/GMRES
- Highly scalable, $O(n_{\text{iter}}N)$ complexity
- Very successful; industrial applications in electromagnetics, acoustics, etc.

But . . .

- What if n_{iter} is large (high contrasts, geometric singularities)?
- What if there are many RHS's (time stepping, inverse problems)?

Compare with direct solvers: no convergence issues, efficient information reuse.

Many hierarchical matrix problems can be solved efficiently using FMM.

- Example: $Ax = b$ using FMM + CG/GMRES
- Highly scalable, $O(n_{\text{iter}} N)$ complexity
- Very successful; industrial applications in electromagnetics, acoustics, etc.

But . . .

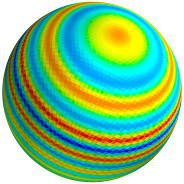
- What if n_{iter} is large (high contrasts, geometric singularities)?
- What if there are many RHS's (time stepping, inverse problems)?

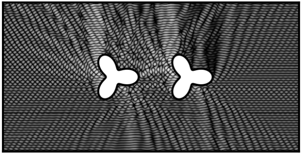
Compare with direct solvers: no convergence issues, efficient information reuse.

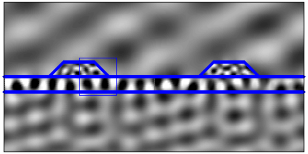
In certain important environments, there is a need for fast direct methods.

Example: wave scattering

- Time-harmonic scattering: Helmholtz equation
- PDE/IE: $A(\Omega)x = b(\theta)$
  - Ω: scatterer geometry/properties
  - θ: angle of incident wave
- Need to analyze response for n_θ incident angles
- Cost: $n_\theta \sim 100$–1000 solves with a fixed matrix

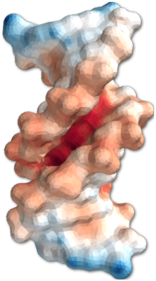




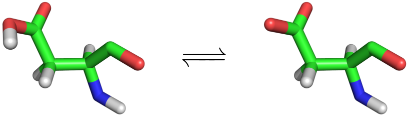


- Extensions: multiple scattering, materials design

Example: protein pK_a calculations

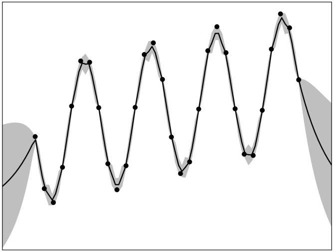


- Electrostatics: linearized Poisson-Boltzmann equation
- PDE/IE: $A(\Omega)x = b(q)$
  - Ω: molecular geometry/properties
  - q: atomic partial charges
- Cost: n_{titr} solves, one for each site to be charged on/off



- Conformational flexibility: $\Omega = \Omega(q)$
- Need local updates, $O((n_{\text{titr}} n_{\text{rot}})^p)$ perturbed solves

Example: uncertainty quantification



- Gaussian process regression
- Observations: (x_0, y_0)
- K: prior covariance kernel
- Posterior prediction: (x, y)
  - $y \sim \mathcal{N}(\mu, \Sigma)$
  - $\mu = K(x, x_0)\,(K_0 + \sigma^2 I)^{-1} y_0$
  - $\Sigma = K_x - K(x, x_0)\,(K_0 + \sigma^2 I)^{-1} K(x_0, x)$

- Extension: online regression, adding new observations
- Conditional sampling: $\hat{y} = \mu + \Sigma^{1/2} z$
- Monte Carlo simulation: n_{samp} RHS's

Overview

- **This talk**: our previous and ongoing work on fast direct matrix methods
- System solvers, least squares, matrix factorizations, updating
- Aim: optimal linear or quasilinear complexity
- Many related contributors: Ambikasaran, Bebendorf, Börm, Bremer, Chandrasekaran, Chen, Corona, Darve, Gillman, Greengard, Gu, Hackbusch, Li, Martinsson, Rokhlin, Schmitz, Starr, Xia, Ying, Young, Zorin
- **Outlook**: almost enough technology to make a deep run at some hard problems

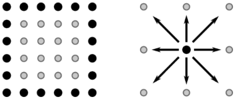
Low-rank compression: interpolative decomposition

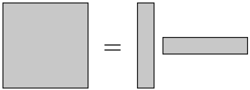
If $A_{:,q}$ is numerically low-rank, then there exist

- skeleton ($\hat{q}$) and redundant ($\check{q}$) columns partitioning $q = \hat{q} \cup \check{q}$
- an interpolation matrix T_q

such that

$$A_{:,\check{q}} \approx A_{:,\hat{q}} T_q \qquad A_{:,(\hat{q},\check{q})} \approx A_{:,\hat{q}} \begin{bmatrix} I & T_q \end{bmatrix}$$





- Essentially a pivoted QR written slightly differently
- Rank-revealing to any specified precison $\epsilon > 0$

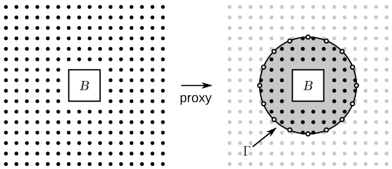
[Cheng/Gimbutas/Martinsson/Rokhlin, Gu/Eisenstat]

Proxy compression

- Algorithms will require IDs of tall-and-skinny matrices of size $O(N)$
- Nominally requires at least $O(N)$ work
- **Observation**: if $A = UV$ then an ID of V gives an ID of A

$$A_{:,\check{q}} = UV_{:,\check{q}} \approx UV_{:,\hat{q}} T_q = A_{:,\hat{q}} T_q$$

- Small V always exists since A is low-rank; how to find V a priori?
- Application-specific:
  - Use Green's theorem/uniqueness of BVP for PDEs
  - Use identity theorem for analytic kernels



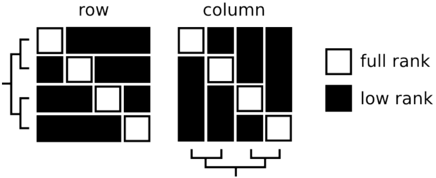
[Cheng/Gimbutas/Martinsson/Rokhlin, Corona/Martinsson/Zorin, Gillman/Young/Martinsson, Greengard/Gueyffier/Martinsson/Rokhlin, Ho/Greengard, Ho/Ying, Martinsson/Rokhlin, Ying, Ying/Biros/Zorin]

Algorithms

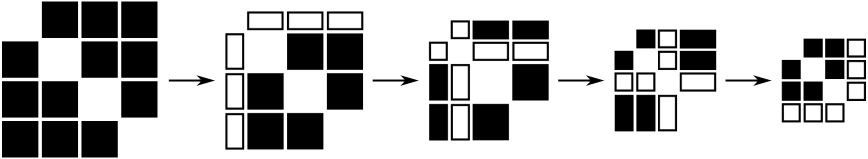
- System solvers, least squares, matrix factorizations, updating
- Focus primarily on elliptic PDEs/IEs

Matrix compression

- Matrix structure: low-rank off-diagonal blocks at each level of a tree hierarchy



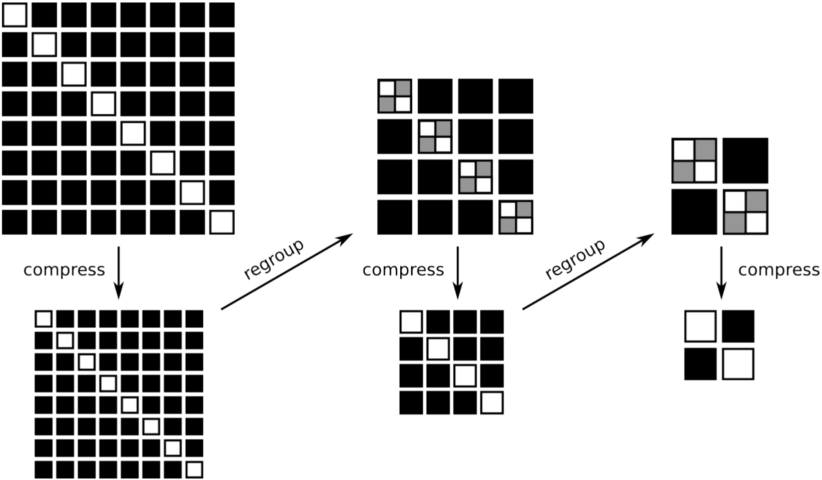
- One-level compression:



- Skeleton "submatrix" has the same structure $\implies$ recurse

[Gillman/Young/Martinsson, Ho/Greengard, Martinsson/Rokhlin]

Matrix compression

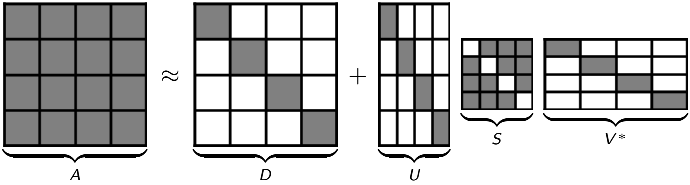


Multilevel compression: recursive skeletonization

[Gillman/Young/Martinsson, Ho/Greengard, Martinsson/Rokhlin]

Matrix compression

- One-level additive decomposition: $A \approx D + USV^*$



- Hierarchical: multilevel telescoping representation

$$A \approx D_0 + U_0(D_1 + U_1(\cdots D_L + U_L S V_L^* \cdots) V_1^*) V_0^*$$

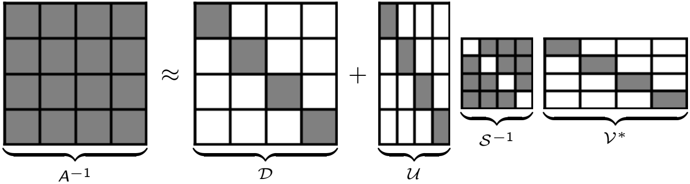
[Gillman/Young/Martinsson, Ho/Greengard, Martinsson/Rokhlin]

Matrix inversion

- **Extended sparsification**: $Ax \approx (D + USV^*)x = b$ is equivalent to

$$\begin{bmatrix} D & U & \\ V^* & & -I \\ & -I & S \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}$$

- Variant of Sherman-Morrison-Woodbury:



- Reduces inversion of (large) A to that of (smaller) S
- Hierarchical: recurse!

[Gillman/Young/Martinsson, Ho/Greengard, Martinsson/Rokhlin]

Matrix inversion

- Extended sparsification:

$$\begin{bmatrix} D_0 & U_0 & & & & & & \\ V_0^* & & -I & & & & & \\ & -I & D_1 & U_1 & & & & \\ & & V_1^* & \ddots & \ddots & & & \\ & & & \ddots & D_L & U_L & & \\ & & & & V_L^* & & -I & \\ & & & & & -I & S \end{bmatrix} \begin{bmatrix} x \\ y_0 \\ z_0 \\ \vdots \\ \vdots \\ y_L \\ z_L \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

- Variant of SMW:

$$\mathcal{A} \approx \mathcal{D}_0 + \mathcal{U}_0(\mathcal{D}_1 + \mathcal{U}_1(\cdots \mathcal{D}_L + \mathcal{U}_L \mathcal{S}^{-1} \mathcal{V}_L^* \cdots)\mathcal{V}_1^*)\mathcal{V}_0^*$$

- Fast direct solver or preconditioner depending on accuracy

[Gillman/Young/Martinsson, Ho/Greengard, Martinsson/Rokhlin]

Theorem

If the off-diagonal block rank is $O(1)$, then the total cost is $O(N)$.

- Optimal for IEs in 1D, PDEs in 2D (after reduction to separators)
- **Method of choice** due to robustness and efficiency
- Applies also to various covariance matrices, other common kernels

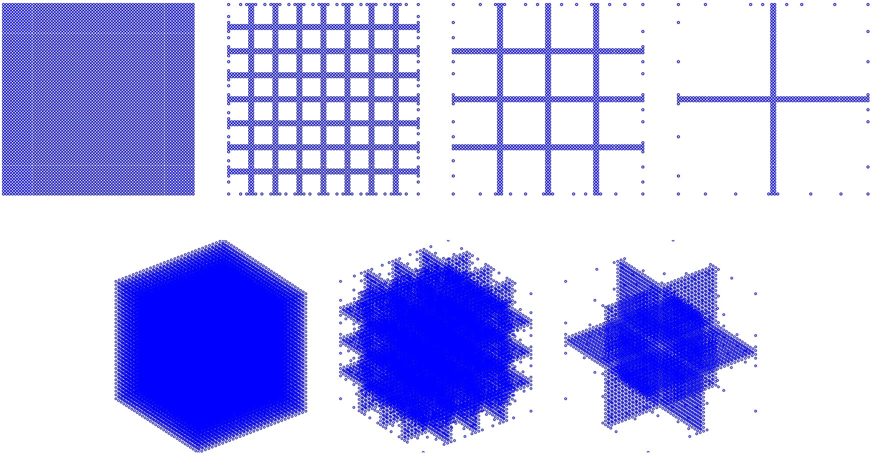
What about IEs in higher dimensions? Multifrontal-like:

	1D	2D	3D
Rank	$O(\log N)$	$O(N^{1/2})$	$O(N^{2/3})$
Precomp	$O(N)$	$O(N^{3/2})$	$O(N^2)$
Solve	$O(N)$	$O(N \log N)$	$O(N^{4/3})$

[Ho/Greengard]

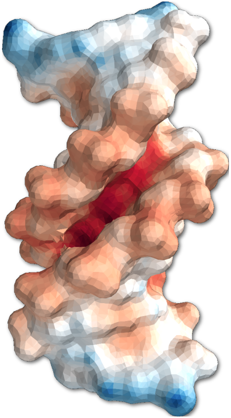
Recursive skeletonization

- Analogous to nested dissection/multifrontal [Duff/Reid, George]



[Ho/Greengard, Ho/Ying]

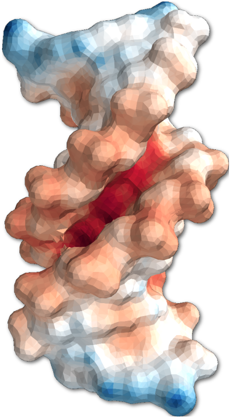
RS for molecular electrostatics



- Computational complexities
  - Precomp: $O(N^{3/2})$
  - Solve: $O(N \log N)$
- Suboptimal but hopefully fast like MF
- DNA system with $N = 20{,}000$, $\epsilon = 10^{-3}$
  - FMM/GMRES: 30 s
  - RS precomp: 10 min
  - RS solve: 0.1 s
- Break-even point: 20 solves
- Effective for small molecules
- Does not scale well to macromolecules ($N \gtrsim 10^6$)

[Ho/Greengard]

RS for molecular electrostatics



- Computational complexities
  - Precomp: $O(N^{3/2})$
  - Solve: $O(N \log N)$
- Suboptimal but hopefully fast like MF
- DNA system with $N = 20{,}000$, $\epsilon = 10^{-3}$
  - FMM/GMRES: 30 s
  - RS precomp: 10 min
  - RS solve: 0.1 s
- Break-even point: 20 solves
- Effective for small molecules
- Does not scale well to macromolecules ($N \gtrsim 10^6$)

How to accelerate to linear complexity?

[Ho/Greengard]

Least squares

- Now suppose that $A \in \mathbb{C}^{M \times N}$ with $M > N$, want to do least squares
- Recall the square case:

$$\begin{bmatrix} D & U & \\ V^* & & -I \\ & -I & S \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}$$

- Variable identities remain, only first row to be interpreted in least squares sense
- Dense LS problem $\min_x \|Ax - b\|$ equivalent to sparse LSE problem

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - b\| \quad \text{s.t.} \quad \mathbf{C}\mathbf{x} = 0$$

$$\mathbf{A} = \begin{bmatrix} D & U & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} V^* & & -I \\ & -I & S \end{bmatrix}$$

- Extended constraints in multilevel setting

[Ho/Greengard]

Least squares

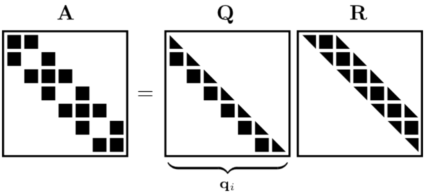
- Solve LSE by weighting + deferred correction (iterative refinement)

$$\min_x \|\mathbf{A}\mathbf{x} - b\| \quad \text{s.t.} \quad \mathbf{C}\mathbf{x} = \mathbf{d}$$

  At each iteration, solve

$$\min_{\mathbf{x}_k} \left\| \begin{bmatrix} \mathbf{A} \\ \tau\mathbf{C} \end{bmatrix} \mathbf{x}_k - \begin{bmatrix} \mathbf{f}_k \\ \tau\mathbf{g}_k \end{bmatrix} \right\|$$

- Fixed matrix, can precompute sparse QR factors
- Semi-direct method, $O(M+N)$ complexity if rank is bounded

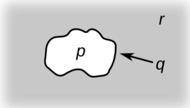


[Ho/Greengard]

Matrix factorization

- Sparse matrices can be factorized/eliminated efficiently

$$A = \begin{bmatrix} A_{pp} & A_{pq} & \\ A_{qp} & A_{qq} & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}$$



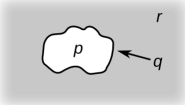
$$R_p^* A S_p = \begin{bmatrix} A_{pp} & & \\ & * & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}, \qquad R_p^* = \begin{bmatrix} I & & \\ * & I & \\ & & I \end{bmatrix}, \quad S_p = \begin{bmatrix} I & * & \\ & I & \\ & & I \end{bmatrix}$$

- DOFs p have been eliminated
- Interactions involving r are unchanged

Matrix factorization

- Sparse matrices can be factorized/eliminated efficiently

$$A = \begin{bmatrix} A_{pp} & A_{pq} & \\ A_{qp} & A_{qq} & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}$$



$$R_p^* A S_p = \begin{bmatrix} A_{pp} & & \\ & * & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}, \qquad R_p^* = \begin{bmatrix} I & & \\ * & I & \\ & & I \end{bmatrix}, \quad S_p = \begin{bmatrix} I & * & \\ & I & \\ & & I \end{bmatrix}$$

- DOFs p have been eliminated
- Interactions involving r are unchanged

How about structured dense matrices?

Matrix factorization

- Let $A = \begin{bmatrix} A_{pp} & A_{pq} \\ A_{qp} & A_{qq} \end{bmatrix}$ with $A_{q\check{p}} \approx A_{q\hat{p}} T_p$ and $A_{\check{p}q} \approx T_p^* A_{\hat{p}q}$
- Reorder $A = \begin{bmatrix} A_{\check{p}\check{p}} & A_{\check{p}\hat{p}} & A_{\check{p}q} \\ A_{\hat{p}\check{p}} & A_{\hat{p}\hat{p}} & A_{\hat{p}q} \\ A_{q\check{p}} & A_{q\hat{p}} & A_{qq} \end{bmatrix}$, define $Q_p = \begin{bmatrix} I & & \\ -T_p & I & \\ & & I \end{bmatrix}$
- **Sparsify** via ID: $Q_p^* A Q_p \approx \begin{bmatrix} * & * & \\ * & A_{\hat{p}\hat{p}} & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix} \xrightarrow{\text{elim}} \begin{bmatrix} * & & \\ & * & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix}$
- Reduces to a subsystem involving skeletons only

[Ho/Ying, Xia/Xi/Gu]

Algorithm: recursive skeletonization factorization

Build tree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
end for

- Block diagonalization:

$$D \approx U_{L-1}^* \cdots U_0^* A V_0 \cdots V_{L-1}$$

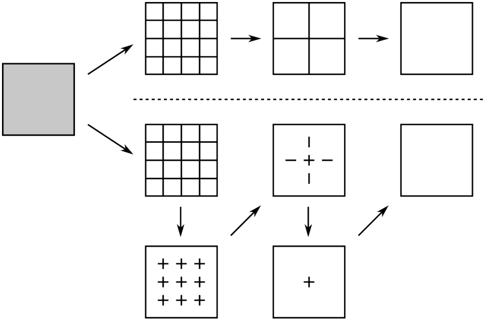
- Generalized LU decomposition:

$$A \approx U_0^{-*} \cdots U_{L-1}^{-*} D V_{L-1}^{-1} \cdots V_0^{-1}$$
$$A^{-1} \approx V_0 \cdots V_{L-1} D^{-1} U_L^* \cdots U_0^*$$

[Ho/Ying]

Accelerating RS for IEs

- RS: $O(N)$ in 1D, $O(N^{3/2})$ in 2D, $O(N^2)$ in 3D
- Superlinear cost in 2D/3D due to skeleton growth
- Skeletons cluster near cell interfaces by Green's theorem
- Exploit skeleton geometry by further skeletonizing along interfaces
- Recursive dimensional reduction



[Corona/Martinsson/Zorin, Ho/Ying, Xia/Chandrasekaran/Gu/Li]

Hierarchical interpolative factorization for IEs in 2D

Build quadtree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
    Let $C_{\ell+1/2}$ be the set of all edges on level ℓ.
    for each cell $c \in C_{\ell+1/2}$ **do**
        Skeletonize remaining DOFs in c.
    end for
end for

[Ho/Ying]

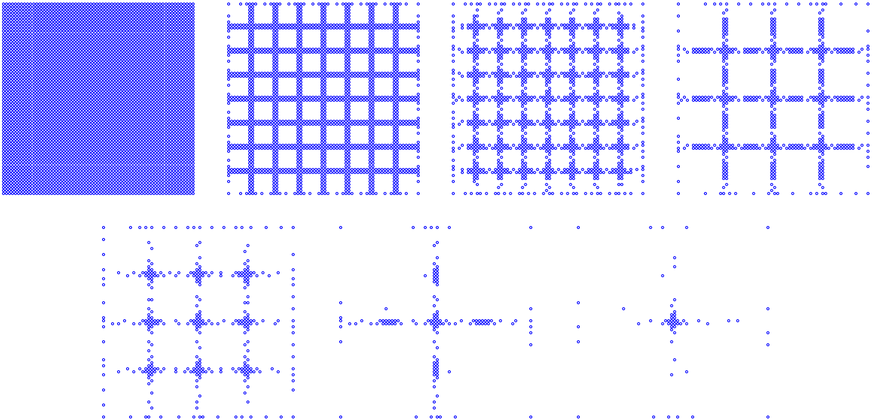
Hierarchical interpolative factorization for IEs in 3D

Build octree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
  Let C_ℓ be the set of all cells on level ℓ.
  for each cell $c \in C_\ell$ **do**
    Skeletonize remaining DOFs in c.
  end for
  Let $C_{\ell+1/3}$ be the set of all faces on level ℓ.
  for each cell $c \in C_{\ell+1/3}$ **do**
    Skeletonize remaining DOFs in c.
  end for
  Let $C_{\ell+2/3}$ be the set of all edges on level ℓ.
  for each cell $c \in C_{\ell+2/3}$ **do**
    Skeletonize remaining DOFs in c.
  end for
end for

[Ho/Ying]

HIF-IE in 2D

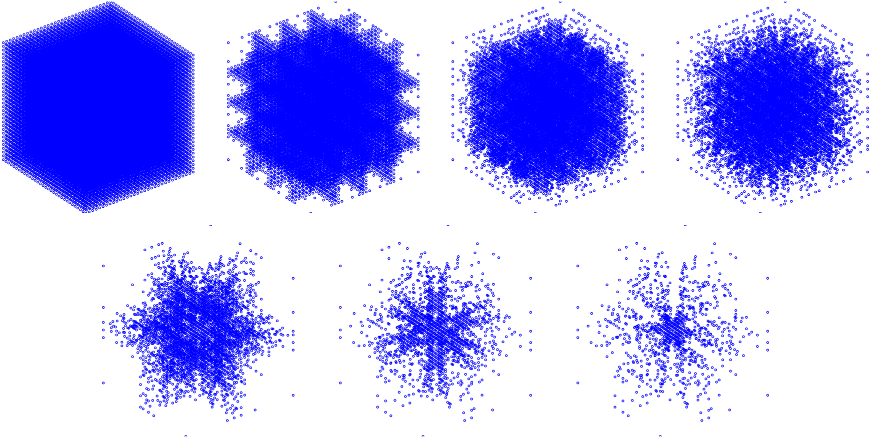
- Skeletonize cells (2D), then edges (1D) hierarchically up a tree



[Ho/Ying]

HIF-IE in 3D

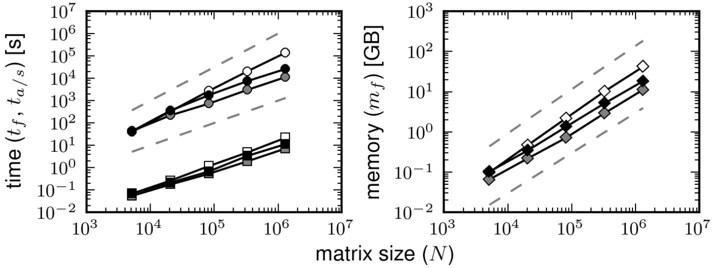
- Skeletonize cells (3D), then faces (2D), then edges (1D) hierarchically up a tree



[Ho/Ying]

HIF-IE results

Second-kind boundary IE for interior Dirichlet Laplace on the unit sphere:



- **rskelf3** (white), **hifie3** (gray), **hifie3x** (black)
- Factorization time (○), solve time (□), memory (◇) at precision $\epsilon = 10^{-3}$
- Reference scalings (gray dashes):
  - Left: $O(N)$ and $O(N^{3/2})$
  - Right: $O(N)$ and $O(N \log N)$

[Ho/Ying]

HIF-IE remarks

- Empirical **linear complexity** for IEs but no proof yet
- Matrix factorization as generalized LU decomposition
  - Fast matrix-vector multiplication (generalized FMM)
  - Fast direct solver at high accuracy, preconditioner otherwise
- Extensions: $A^{1/2}$, $\log \det A$, $\operatorname{diag} A^{-1}$
- Modification for sparse PDEs based on MF (HIF-DE)
- Highly parallelizable [with A. Benson, Y. Li, J. Poulson, L. Ying]
- MATLAB codes freely available at `https://github.com/klho/FLAM/`

Updating

- Direct methods: very efficient for a **fixed** matrix with multiple RHS's
- Can accommodate local perturbations using augmented system approach

$$Ax = b \quad \rightarrow \quad \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

- Reuse factorization via SMW or A^{-1} as preconditioner
- Cost: $O(kN)$, where k is perturbation rank or iterations required

[Greengard/Gueyffier/Martinsson/Rokhlin]

Updating

- Direct methods: very efficient for a **fixed** matrix with multiple RHS's
- Can accommodate local perturbations using augmented system approach

$$Ax = b \quad \to \quad \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

- Reuse factorization via SMW or A^{-1} as preconditioner
- Cost: $O(kN)$, where k is perturbation rank or iterations required

What about a sequence of local updates?

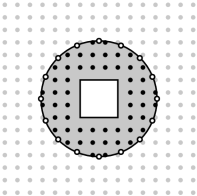
- Works only if all perturbed systems are "close" to a base system
- Cannot accumulate in a global way

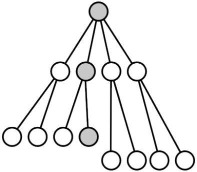
[Greengard/Gueyffier/Martinsson/Rokhlin]

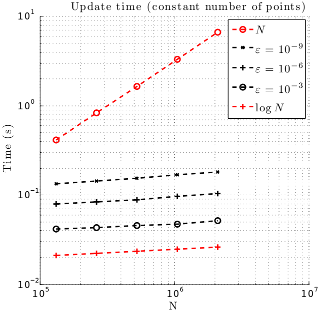
Updating

Another idea: directly update factorization [with A. Damle, V. Minden, L. Ying]

- Use Green's theorem to localize effect of perturbation
- Redo computation up only one branch of the tree: $O(\log N)$ cost







Summary

What we know how to do:

- $O(N)$ factorizations/solvers for IEs and PDEs
- $O(\log N)$ local updates
- Semi-direct least squares

What we don't know how to do (fully):

- How to make small global updates?
- How to form spectral decompositions?
- How to compute matrix functions?

References

- K.L. Ho, L. Greengard. A fast direct solver for structured linear systems by recursive skeletonization. SIAM J. Sci. Comput. 34 (5): A2507–A2532, 2012.
- K.L. Ho, L. Greengard. A fast semidirect least squares algorithm for hierarchically block separable matrices. SIAM J. Matrix Anal. Appl. 35 (2): 725–748, 2014.
- K.L. Ho, L. Ying. Hierarchical interpolative factorization for elliptic operators: differential equations. Preprint, arXiv:1307.2895 [math.NA], 2013.
- K.L. Ho, L. Ying. Hierarchical interpolative factorization for elliptic operators: integral equations. Preprint, arXiv:1307.2666 [math.NA], 2013.