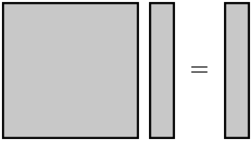
Fast direct methods for structured matrices

Kenneth L. Ho (Stanford)

Caltech, Nov. 2014

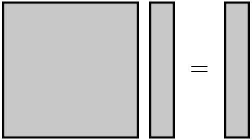
Introduction

$$Ax = b$$



Introduction

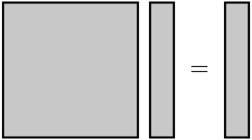
$$Ax = b$$



- For $A \in \mathbb{C}^{N \times N}$ dense, solution generally requires $O(N^3)$ work
- Classical methods infeasible beyond $N \sim 10^4$

Introduction

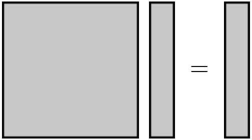
$$Ax = b$$



- For $A \in \mathbb{C}^{N \times N}$ dense, solution generally requires $O(N^3)$ work
- Classical methods infeasible beyond $N \sim 10^4$
- Other common matrix problems:
  - $y = Ax$: $O(N^2)$
  - $A = UV^*$: $O(N^3)$
  - $\Delta = \det A$: $O(N^3)$

Introduction

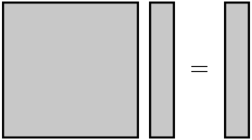
$$Ax = b$$



- For $A \in \mathbb{C}^{N \times N}$ dense, solution generally requires $O(N^3)$ work
- Classical methods infeasible beyond $N \sim 10^4$
- Other common matrix problems:
  - $y = Ax$: $O(N^2)$
  - $A = UV^*$: $O(N^3)$
  - $\Delta = \det A$: $O(N^3)$
- Observation: many matrices arising in practice are **structured**

Introduction

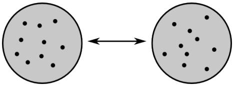
$$Ax = b$$

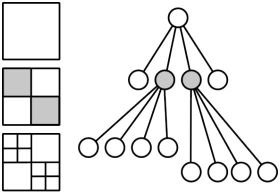


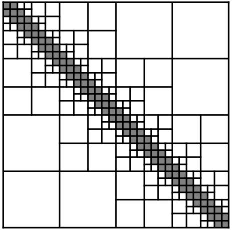
- For $A \in \mathbb{C}^{N \times N}$ dense, solution generally requires $O(N^3)$ work $\to O(N)$
- Classical methods infeasible beyond $N \sim 10^4$
- Other common matrix problems:
  - $y = Ax$: $O(N^2) \to O(N)$
  - $A = UV^*$: $O(N^3) \to O(N)$
  - $\Delta = \det A$: $O(N^3) \to O(N)$
- Observation: many matrices arising in practice are structured
- Goal: accelerate to **linear complexity** by exploiting matrix structure

Introduction

- **Hierarchical matrices**: low-rank submatrices at a hierarchy of scales
- Canonical example: N-body problem
  - Particle locations: $x_i,\ i = 1, \ldots, N$
  - 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



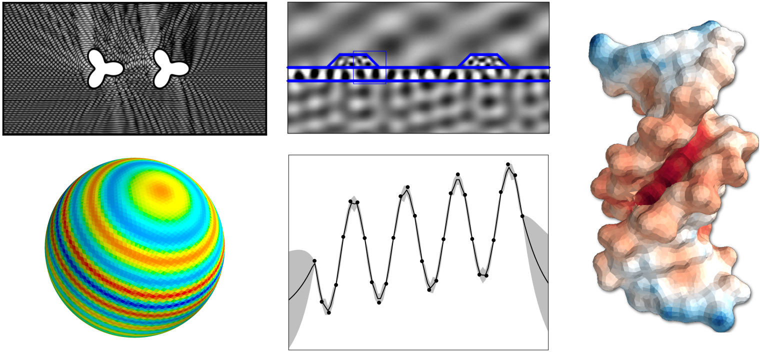




[Greengard/Rokhlin 1987, Hackbusch/Börm 2002]

Introduction

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



[Greengard/Ho/Lee 2014, Ho 2012, Ho/Greengard 2012]

Many structured matrix problems can be solved efficiently by iteration

- CG/GMRES + fast multiplication: $O(n_{\text{iter}} N)$ complexity
- Very successful; industrial applications in electromagnetics, acoustics, etc.

Introduction

Many structured matrix problems can be solved efficiently by iteration

- CG/GMRES + fast multiplication: $O(n_{\mathrm{iter}} N)$ complexity
- Very successful; industrial applications in electromagnetics, acoustics, etc.

But . . .

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

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

Many structured matrix problems can be solved efficiently by iteration

- CG/GMRES + fast multiplication: $O(n_{\text{iter}} N)$ complexity
- Very successful; industrial applications in electromagnetics, acoustics, etc.

But ...

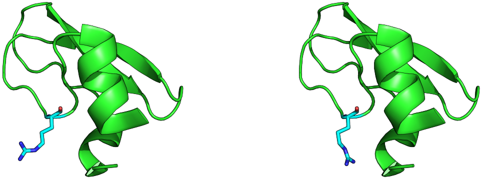
- What if n_{iter} is large (high contrasts, geometric singularities, ill-conditioning)?
- What if there are many RHS's (time stepping, inverse problems)?

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

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

Example: protein design

- Protein defined by a fixed backbone with flexible residue sidechains
- Each sidechain can be one of several rotamers $r_i \in R_i$
- Energy $E(\mathbf{r})$ depends on the joint rotamer configuration $\mathbf{r}$
- Goal: find $\mathbf{r}$ such that $E(\mathbf{r})$ is minimized



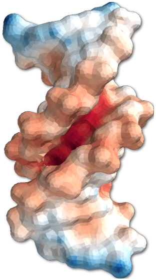
- NP-hard but various strategies are available
- One of many related formulations

[Pierce/Winfree 2002]

Example: protein design

- Simplest approach: pairwise approximation

$$E(\mathbf{r}) \approx \sum_i E(r_i) + \frac{1}{2} \sum_i \sum_{j \neq i} E(r_i, r_j)$$



- Number of energy evaluations: $O((n_{\text{rot}} N_{\text{res}})^2)$
- Each evaluation requires a PDE solve for the electrostatic energy:

$$A_i x_i = b_i, \quad i = 1, \ldots, O((n_{\text{rot}} N_{\text{res}})^2)$$

- Matrices A_i are perturbations of fixed backbone matrix A_0
- Precompute A_0^{-1}, rapid update for each $x_i = A_i^{-1} b_i$

Potential for massive acceleration using fast direct methods.

[Marshall/Vizcarra/Mayo 2005]

Overview

- **This talk**: our recent work on fast direct methods for structured matrices
- Many other contributors (apologies for an incomplete list)
- Focus on integral equations in 2D/3D, complex geometry
- Main result: linear-complexity generalized LU decomposition
- Sparsification/elimination + recursive dimensional reduction

[Ambikasaran, Bebendorf, Börm, Bremer, Chandrasekaran, Chen, Corona, Darve, Gillman, Greengard, Gu, Hackbusch, Li, Martinsson, Rokhlin, Schmitz, Starr, Xia, Ying, Young, Zorin, . . .]

Overview

- **This talk**: our recent work on fast direct methods for structured matrices
- Many other contributors (apologies for an incomplete list)
- Focus on integral equations in 2D/3D, complex geometry
- Main result: linear-complexity generalized LU decomposition
- Sparsification/elimination + recursive dimensional reduction

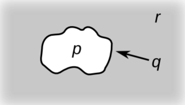
Tools: sparse elimination, interpolative decomposition, skeletonization

[Ambikasaran, Bebendorf, Börm, Bremer, Chandrasekaran, Chen, Corona, Darve, Gillman, Greengard, Gu, Hackbusch, Li, Martinsson, Rokhlin, Schmitz, Starr, Xia, Ying, Young, Zorin, . . .]

Sparse elimination

Let

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



(Think of A as a sparse matrix.) If A_{pp} is nonsingular, define

$$R_p^* = \begin{bmatrix} I & & \\ -A_{qp}A_{pp}^{-1} & I & \\ & & I \end{bmatrix}, \quad S_p = \begin{bmatrix} I & -A_{pp}^{-1}A_{pq} & \\ & I & \\ & & I \end{bmatrix}$$

so that

$$R_p^* A S_p = \begin{bmatrix} A_{pp} & & \\ & * & A_{qr} \\ & A_{rq} & A_{rr} \end{bmatrix}.$$

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

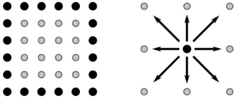
Interpolative decomposition

If $A_{:,q}$ has numerical rank k, then there exist

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

such that

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



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

Interactions between separated regions are low-rank.

[Cheng/Gimbutas/Martinsson/Rokhlin 2005, Gu/Eisenstat 1996]

Skeletonization

- Efficient elimination of redundant DOFs
- Let $A = \begin{bmatrix} A_{pp} & A_{pq} \\ A_{qp} & A_{qq} \end{bmatrix}$ with A_{pq} and A_{qp} low-rank
- Apply ID to $\begin{bmatrix} A_{qp} \\ A_{pq}^* \end{bmatrix}$: $\quad \begin{bmatrix} A_{q\check{p}} \\ A_{\check{p}q}^* \end{bmatrix} \approx \begin{bmatrix} A_{q\hat{p}} \\ A_{\hat{p}q}^* \end{bmatrix} T_p \implies \begin{array}{c} A_{q\check{p}} \approx A_{q\hat{p}} T_p \\ A_{\check{p}q} \approx T_p^* A_{\hat{p}q} \end{array}$
- 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 2013, Xia/Xi/Gu 2012]

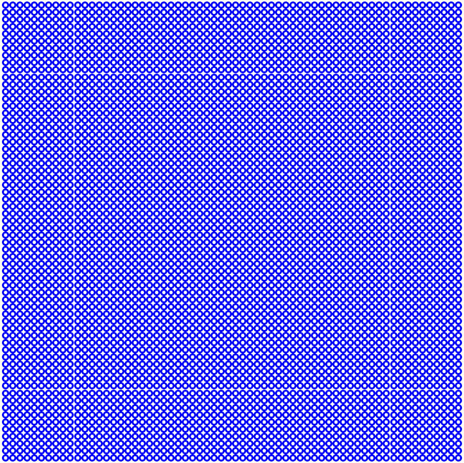
Algorithm: recursive skeletonization factorization

Build quadtree/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
end for

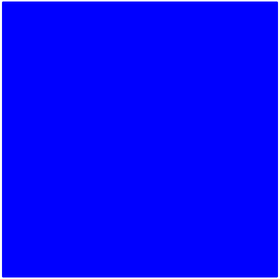
- Reformulation of old algorithm using new elimination framework

[Gillman/Young/Martinsson 2012, Ho/Greengard 2012, Ho/Ying 2013, Martinsson/Rokhlin 2005]

RSF in 2D: level 0



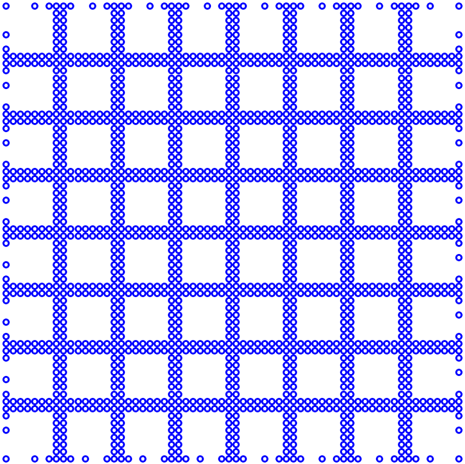
domain



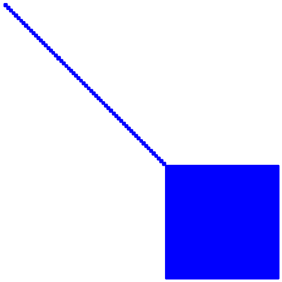
matrix

[Ho/Ying 2013]

RSF in 2D: level 1



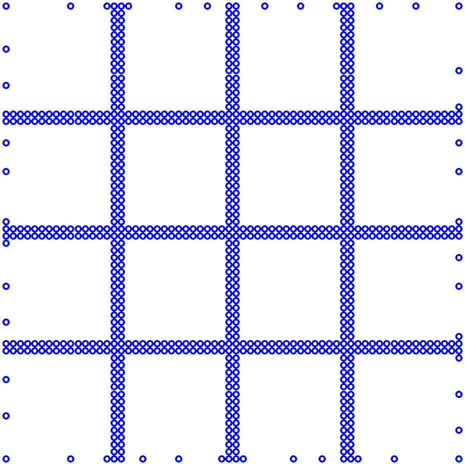
domain



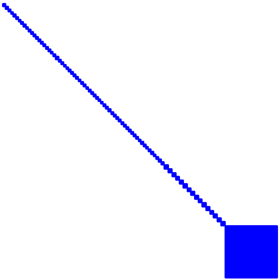
matrix

[Ho/Ying 2013]

RSF in 2D: level 2



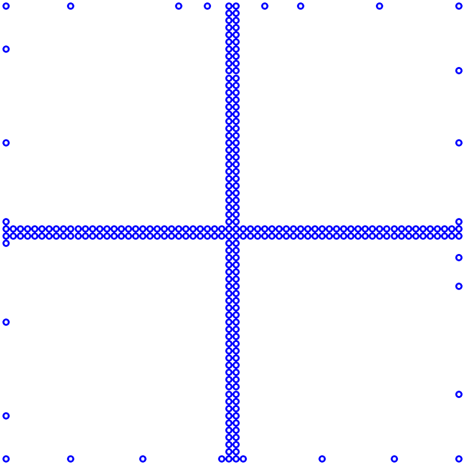
domain



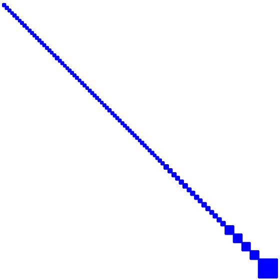
matrix

[Ho/Ying 2013]

RSF in 2D: level 3



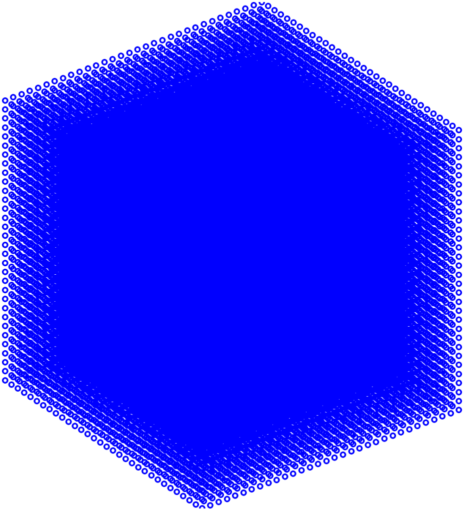
domain



matrix

[Ho/Ying 2013]

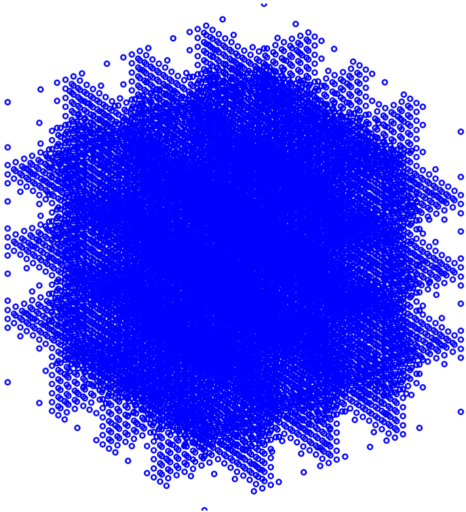
RSF in 3D: level 0



domain

[Ho/Ying 2013]

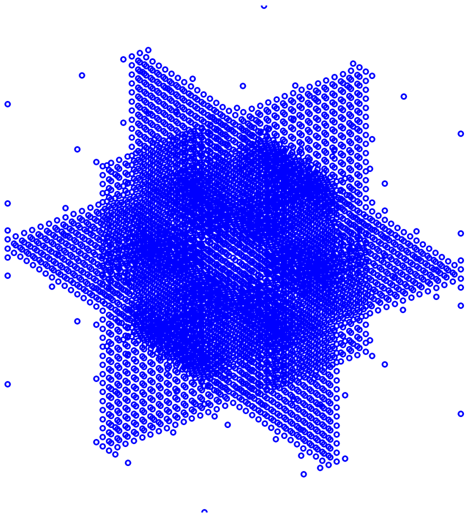
RSF in 3D: level 1



domain

[Ho/Ying 2013]

RSF in 3D: level 2



domain

[Ho/Ying 2013]

RSF analysis

- Skeletonization operators:

$$U_\ell = \prod_{c \in C_\ell} Q_c R_{\check{c}}, \quad V_\ell = \prod_{c \in C_\ell} Q_c S_{\check{c}}$$

$$Q_c = \begin{bmatrix} I & & \\ * & I & \\ & & I \end{bmatrix}, \quad R_{\check{c}}, S_{\check{c}} = \begin{bmatrix} I & * & \\ & I & \\ & & I \end{bmatrix}$$

- Block diagonalization:

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

- Generalized LU decomposition:

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

- Fast direct solver or preconditioner

[Ho/Ying 2013]

RSF analysis

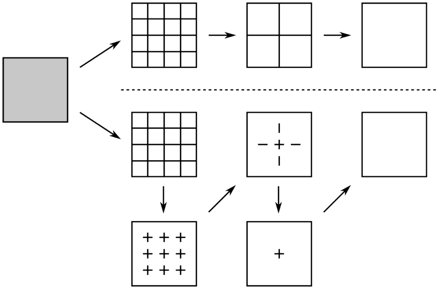
The cost is determined by the skeleton size.

	1D	2D	3D
Skeleton size	$O(\log N)$	$O(N^{1/2})$	$O(N^{2/3})$
Factorization cost	$O(N)$	$O(N^{3/2})$	$O(N^2)$
Solve cost	$O(N)$	$O(N \log N)$	$O(N^{4/3})$

Question: How to reduce the skeleton size in 2D and 3D?

[Ho/Greengard 2012, Martinsson/Rokhlin 2005]

RSF analysis



Question: How to reduce the skeleton size in 2D and 3D?

- Skeletons cluster near cell interfaces (Green's theorem)
- Exploit skeleton geometry by further skeletonizing along interfaces
- Dimensional reduction

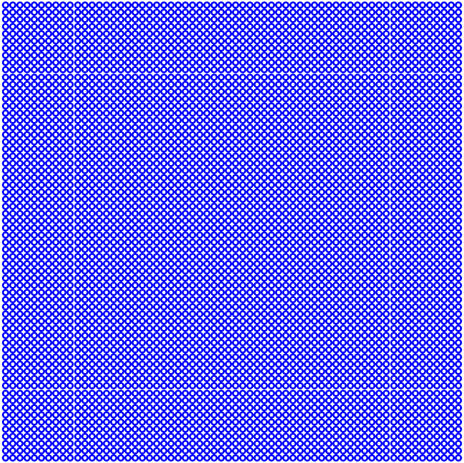
[Corona/Martinsson/Zorin 2013, Ho/Ying 2013, Xia/Chandrasekaran/Gu/Li 2009]

Algorithm: hierarchical interpolative factorization for IEs in 2D

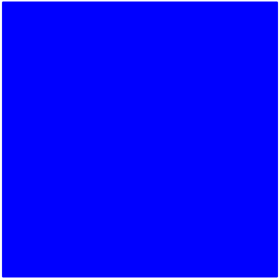
Build quadtree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
    Let $C_{\ell+1/2}$ be the set of all edges on level ℓ.
    for each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

[Ho/Ying 2013]

HIF-IE in 2D: level 0



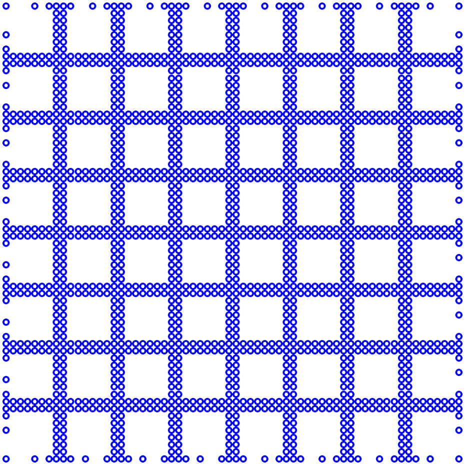
domain



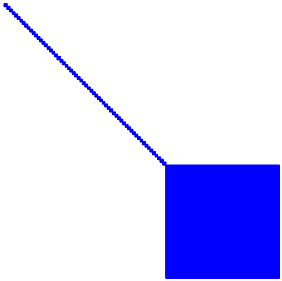
matrix

[Ho/Ying 2013]

HIF-IE in 2D: level 1/2



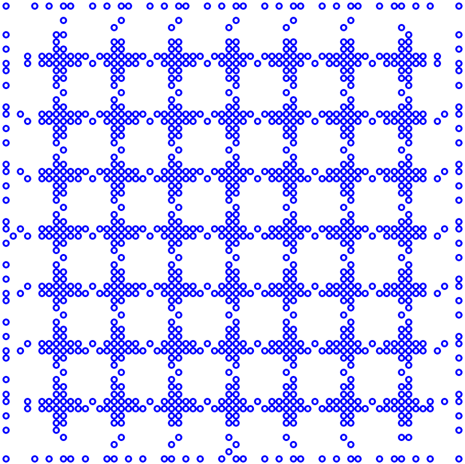
domain



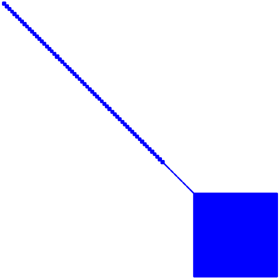
matrix

[Ho/Ying 2013]

HIF-IE in 2D: level 1



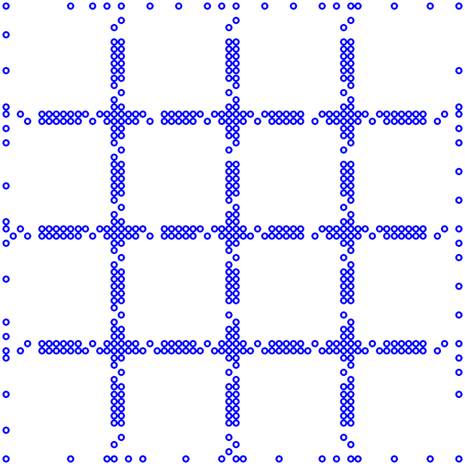
domain



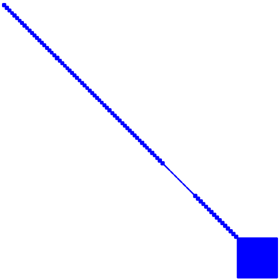
matrix

[Ho/Ying 2013]

HIF-IE in 2D: level 3/2



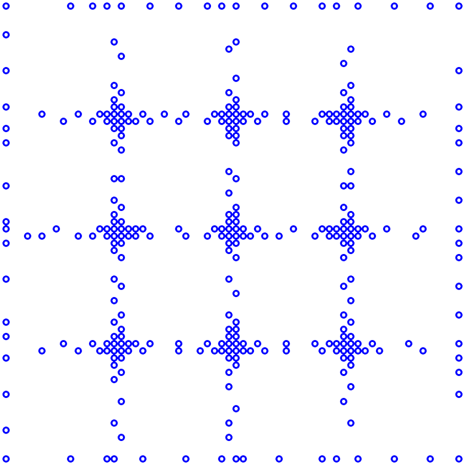
domain



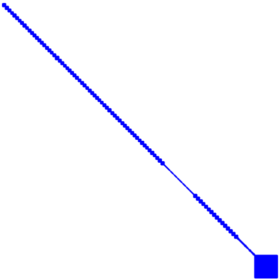
matrix

[Ho/Ying 2013]

HIF-IE in 2D: level 2



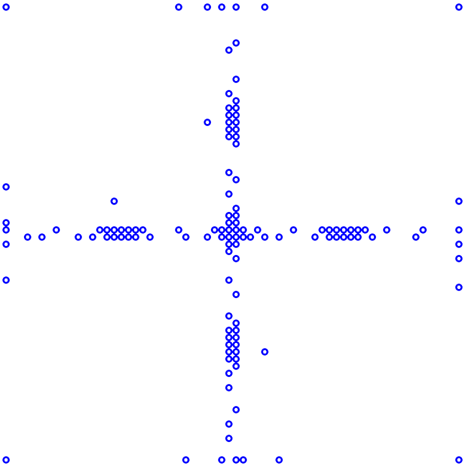
domain



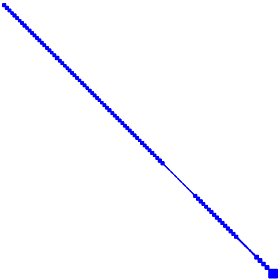
matrix

[Ho/Ying 2013]

HIF-IE in 2D: level 5/2



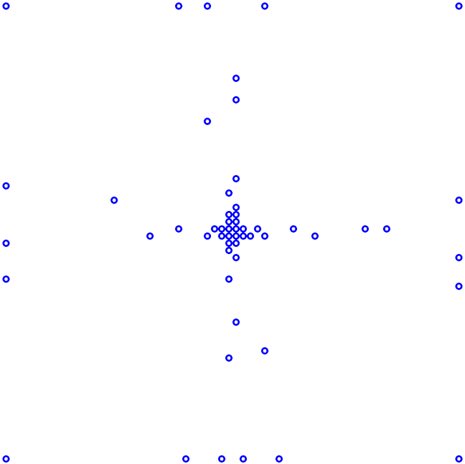
domain



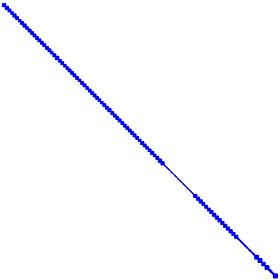
matrix

[Ho/Ying 2013]

HIF-IE in 2D: level 3



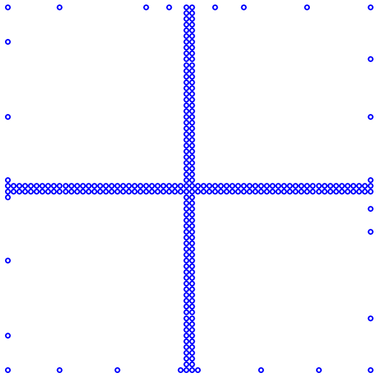
domain



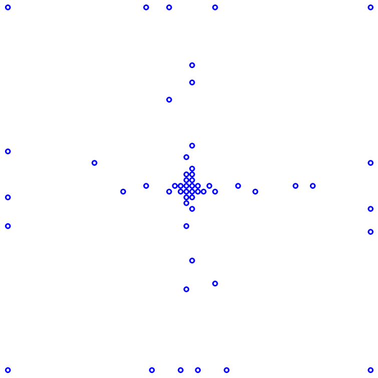
matrix

[Ho/Ying 2013]

RSF vs. HIF-IE in 2D



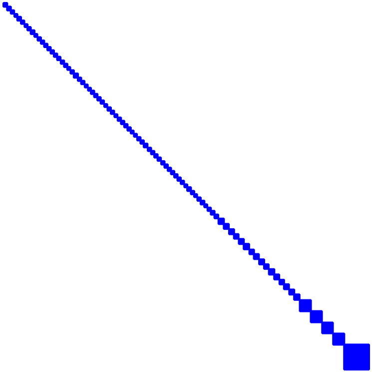
RSF



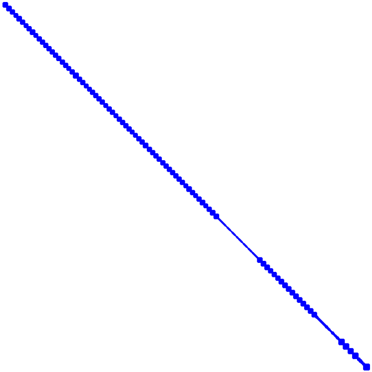
HIF-IE

[Ho/Ying 2013]

RSF vs. HIF-IE in 2D



RSF



HIF-IE

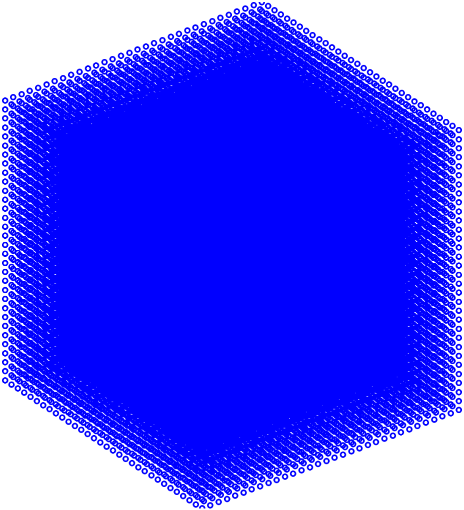
[Ho/Ying 2013]

Algorithm: 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 2013]

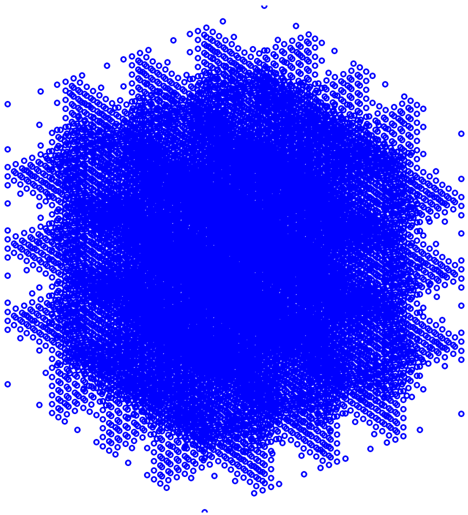
HIF-IE in 3D: level 0



domain

[Ho/Ying 2013]

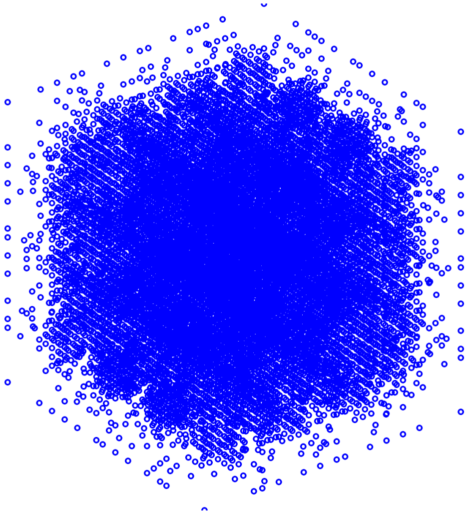
HIF-IE in 3D: level 1/3



domain

[Ho/Ying 2013]

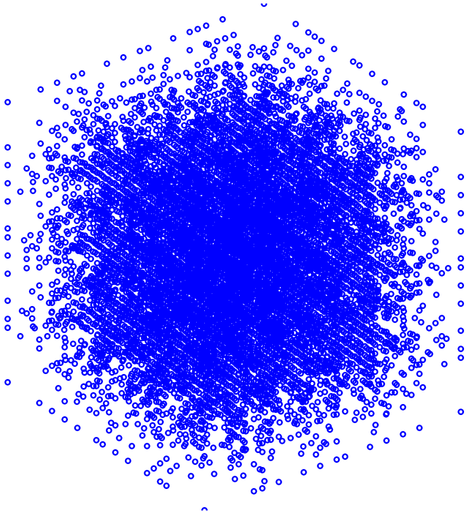
HIF-IE in 3D: level 2/3



domain

[Ho/Ying 2013]

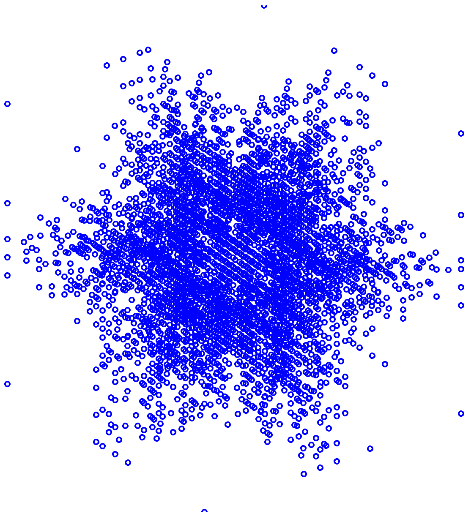
HIF-IE in 3D: level 1



domain

[Ho/Ying 2013]

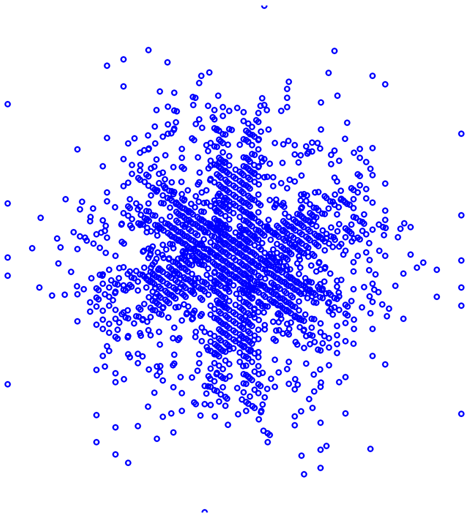
HIF-IE in 3D: level 4/3



domain

[Ho/Ying 2013]

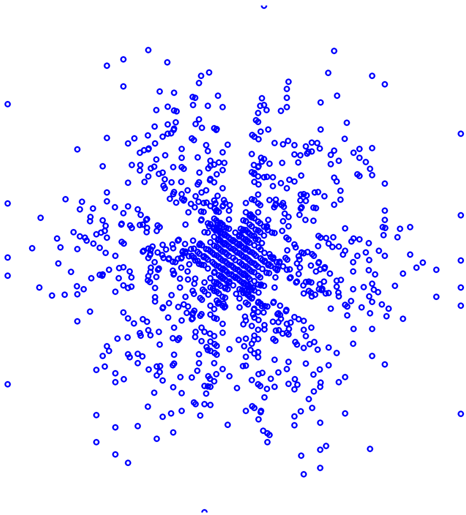
HIF-IE in 3D: level 5/3



domain

[Ho/Ying 2013]

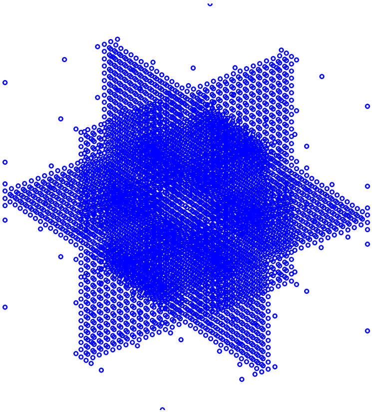
HIF-IE in 3D: level 2



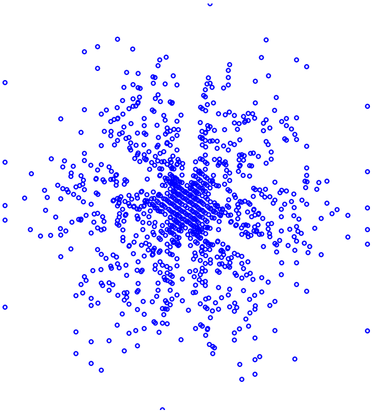
domain

[Ho/Ying 2013]

RSF vs. HIF-IE in 3D



RSF



HIF-IE

[Ho/Ying 2013]

HIF-IE analysis

- 2D:

$$A \approx U_0^{-*} U_{1/2}^{-*} \cdots U_{L-1/2}^{-*} D V_{L-1/2}^{-1} \cdots V_{1/2}^{-1} V_0^{-1}$$
$$A^{-1} \approx V_0 V_{1/2} \cdots V_{L-1/2} D^{-1} U_{L-1/2}^{*} \cdots U_{1/2}^{*} U_0^{*}$$

- 3D:

$$A \approx U_0^{-*} U_{1/3}^{-*} U_{2/3}^{-*} \cdots U_{L-1/3}^{-*} D V_{L-1/3}^{-1} \cdots V_{2/3}^{-1} V_{1/3}^{-1} V_0^{-1}$$
$$A^{-1} \approx V_0 V_{1/3} V_{2/3} \cdots V_{L-1/3} D^{-1} U_{L-1/3}^{*} \cdots U_{2/3}^{*} U_{1/3}^{*} U_0^{*}$$

Conjecture:

Skeleton size:	$O(\log N)$
Factorization cost:	$O(N)$
Solve cost:	$O(N)$

[Ho/Ying 2013]

HIF-IE analysis

- 2D:
$$A \approx U_0^{-*} U_{1/2}^{-*} \cdots U_{L-1/2}^{-*} D V_{L-1/2}^{-1} \cdots V_{1/2}^{-1} V_0^{-1}$$
$$A^{-1} \approx V_0 V_{1/2} \cdots V_{L-1/2} D^{-1} U_{L-1/2}^{*} \cdots U_{1/2}^{*} U_0^{*}$$

- 3D:
$$A \approx U_0^{-*} U_{1/3}^{-*} U_{2/3}^{-*} \cdots U_{L-1/3}^{-*} D V_{L-1/3}^{-1} \cdots V_{2/3}^{-1} V_{1/3}^{-1} V_0^{-1}$$
$$A^{-1} \approx V_0 V_{1/3} V_{2/3} \cdots V_{L-1/3} D^{-1} U_{L-1/3}^{*} \cdots U_{2/3}^{*} U_{1/3}^{*} U_0^{*}$$

Conjecture:

Skeleton size:	$O(\log N)$
Factorization cost:	$O(N)$
Solve cost:	$O(N)$

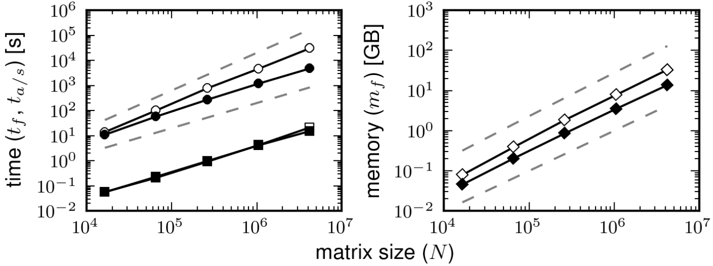
Actually slightly more complicated . . .

[Ho/Ying 2013]

Numerical results in 2D

First-kind volume IE on the unit square:

$$-\frac{1}{2\pi}\int_{(0,1)^2} \log \|x-y\| u(y)\, dA(y) = f(x)$$



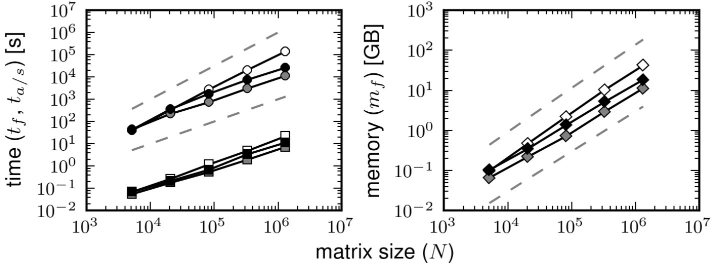
- **rskelf2** (white), **hifie2** (black)
- Factorization time (○), solve time (□), memory (◇) at precision $\epsilon = 10^{-6}$
- Reference scalings (gray dashes):
  - Left: $O(N)$ and $O(N^{3/2})$
  - Right: $O(N)$ and $O(N \log N)$

[Ho/Ying 2013]

Numerical results in 3D

Second-kind boundary IE on the unit sphere:

$$-\frac{1}{2}u(x) + \frac{1}{4\pi}\int_{S^2} \frac{\partial}{\partial \nu(y)}\left(\frac{1}{\|x-y\|}\right) u(y)\, dS(y) = f(x)$$



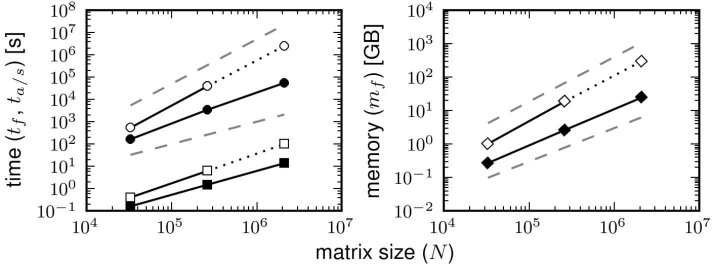
- **rskelf3** (white), **hifie3** (gray), **hifie3x** (black)
- Factorization time (○), solve time (□), memory (◇) at precision $\epsilon = 10^{-3}$
- Reference scalings (gray dashes):
  - Left: $O(N)$ and $O(N^{3/2})$
  - Right: $O(N)$ and $O(N \log N)$

[Ho/Ying 2013]

Numerical results in 3D

First-kind volume IE on the unit cube:

$$\frac{1}{4\pi}\int_{(0,1)^3}\frac{u(y)}{\|x-y\|}\,dV(y)=f(x)$$

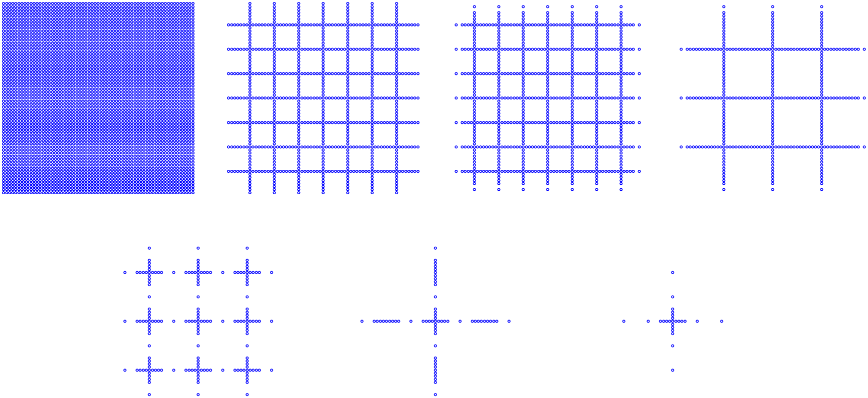


- **rskelf3** (white), **hifie3** (black)
- Factorization time (○), solve time (□), memory (◇) at precision $\epsilon = 10^{-3}$
- Reference scalings (gray dashes):
  - Left: $O(N)$ and $O(N^2)$
  - Right: $O(N)$ and $O(N^{4/3})$

[Ho/Ying 2013]

Hierarchical interpolative factorization for PDEs in 2D

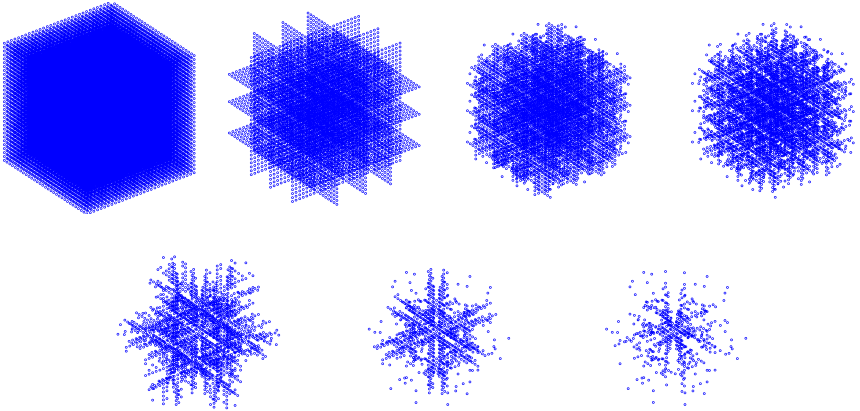
- Build on top of multifrontal to exploit sparsity



[Ho/Ying 2013]

Hierarchical interpolative factorization for PDEs in 3D

- Build on top of multifrontal to exploit sparsity

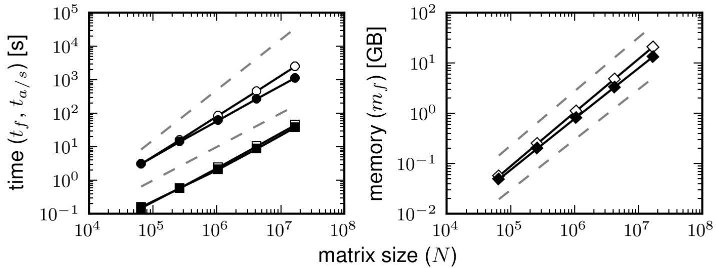


[Ho/Ying 2013]

Numerical results in 2D

Five-point stencil on the unit square with $a(x) = 1$:

$$-\nabla \cdot (a(x) \nabla u(x)) = f(x)$$



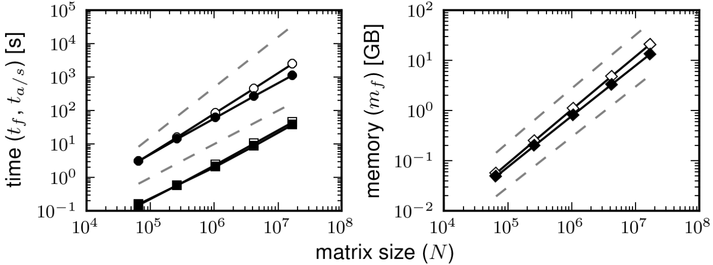
- **mf2** (white), **hifde2** (black)
- Factorization time ($\circ$), solve time ($\square$), memory ($\diamond$) at precision $\epsilon = 10^{-9}$
- Reference scalings (gray dashes):
  - Left: $O(N)$ and $O(N^{3/2})$
  - Right: $O(N)$ and $O(N \log N)$

[Ho/Ying 2013]

Numerical results in 2D

Five-point stencil on the unit square with $a(x)$ a quantized high-contrast ($\kappa \sim 10^4$) random field:

$$-\nabla \cdot (a(x) \nabla u(x)) = f(x)$$



- **mf2** (white), **hifde2** (black)
- Factorization time (○), solve time (□), memory (◇) at precision $\epsilon = 10^{-9}$
- Reference scalings (gray dashes):
  - Left: $O(N)$ and $O(N^{3/2})$
  - Right: $O(N)$ and $O(N \log N)$

[Ho/Ying 2013]

Remarks on HIF

- Efficient factorization of structured operators in 2D/3D
- Empirical linear complexity but no proof yet
- Approximate generalized LU decomposition
  - Fast direct solver or preconditioner
  - Extremely effective for multiple RHS's
- **Extensions**: $A^{1/2}$, $\det A$, $\operatorname{diag} A^{-1}$
- Highly parallelizable [with A. Benson, Y. Li, J. Poulson, L. Ying]
- MATLAB codes available at `https://github.com/klho/FLAM/`

Remarks on HIF

- Efficient factorization of structured operators in 2D/3D
- Empirical linear complexity but no proof yet
- Approximate generalized LU decomposition
  - Fast direct solver or preconditioner
  - Extremely effective for multiple RHS's
- **Extensions**: $A^{1/2}$, $\det A$, $\operatorname{diag} A^{-1}$
- Highly parallelizable [with A. Benson, Y. Li, J. Poulson, L. Ying]
- MATLAB codes available at `https://github.com/klho/FLAM/`

- *Perspective*: structured dense matrices can be sparsified very efficiently
- Can borrow directly from sparse algorithms, e.g., RSF = MF
- What other features of sparse matrices can be exploited?

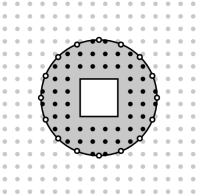
Local updating

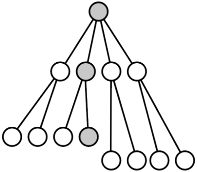
"Naive" approach: local geometric perturbations as low-rank updates

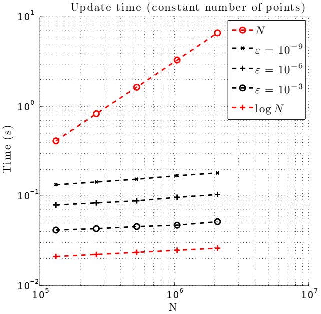
- Sherman-Morrison-Woodbury: rank $k \implies O(Nk)$ cost
- Cannot accumulate updates across domain

Factorization updating [with V. Minden, A. Damle, L. Ying]:

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







[Greengard/Gueyffier/Martinsson/Rokhlin 2009]

Conclusion

- Main thrust of my work: building technology for structured matrices
- Fast multiplication, direct solvers, least squares, factorizations
- Supporting tools: e.g., local updating
- **Outlook**: almost enough technology to make a deep run at some hard problems

Conclusion

- Main thrust of my work: building technology for structured matrices
- Fast multiplication, direct solvers, least squares, factorizations
- Supporting tools: e.g., local updating
- **Outlook**: almost enough technology to make a deep run at some hard problems

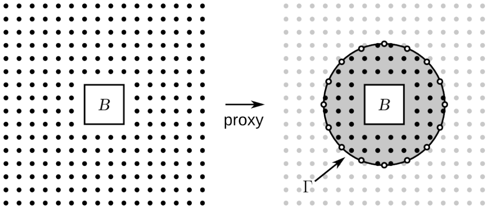
- Other related work:
  - Skeletonization/elimination as adaptive numerical coarsening
  - Butterfly algorithms for oscillatory kernels [with Y. Li, H. Yang, L. Ying]
- Next steps:
  - Global updating, spectral decompositions, matrix functions
  - Applications: biology, materials science, data analysis, UQ

References

- L. Greengard, K.L. Ho, J.-Y. Lee. A fast direct solver for scattering from periodic structures with multiple material interfaces in two dimensions. J. Comput. Phys. 258: 738–751, 2014.
- K.L. Ho. Fast direct methods for molecular electrostatics. Ph.D. thesis, New York University, 2012.
- 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.

Proxy compression

- Main cost of algorithm: computing IDs of tall-and-skinny matrices
- **Global** operation can be reduced to local operation using Green's theorem
- Suffices to compress against neighbors plus "proxy" surface
- Crucial for overcoming $O(N^2)$ complexity



[Cheng/Gimbutas/Martinsson/Rokhlin 2005, Gillman/Young/Martinsson 2012, Ho/Greengard 2012, Ho/Ying 2013, Martinsson/Rokhlin 2005, Ying/Biros/Zorin 2004]

Second-kind IEs

- IEs of the form $a(x)u(x) + \int_\Omega K(x,y)u(y)\,d\Omega(y) = f(x)$
- High contrast in diagonal vs. off-diagonal entries
- Mixing of cell, face, edge in HIF-IE leads to error
- Need to use effective precision $O(\epsilon/N)$
- Quasilinear complexity estimates:

	2D	3D
Factorization cost	$O(N \log N)$	$O(N \log^6 N)$
Solve cost	$O(N \log \log N)$	$O(N \log^2 N)$

[Ho/Ying 2013]