Efficient operator factorizations for integral and differential equations

Kenneth L. Ho (Stanford)

Joint work with Lexing Ying

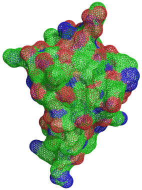
FACM 2014, NJIT

Introduction

Elliptic PDEs in integral or differential form:

$$a(x)u(x) + \int_\Omega K(x,y)u(y)\, d\Omega(y) = f(x)$$
$$-\nabla \cdot (a(x)\nabla u(x)) + b(x)u(x) = f(x)$$

- Fundamental to physics and engineering
- Interested in 2D/3D, complex geometry
- Discretize $\rightarrow$ structured linear system $Au = f$



Goal: fast and accurate algorithms for the discrete operators

- Fast matrix-vector multiplication, fast direct solver, good preconditioner
- Ideally, fast matrix **factorization**
- Linear or nearly linear complexity, high practical efficiency

Direct vs. iterative solvers

- Direct solvers: no iteration (e.g., Gaussian elimination)
- Why direct solvers? Compare with iterative methods

Direct vs. iterative solvers

- Direct solvers: no iteration (e.g., Gaussian elimination)
- Why direct solvers? Compare with iterative methods

Iterative solvers

- GMRES, CG, relaxation methods, multigrid, etc.
- Can achieve linear complexity under certain conditions
- But number of iterations can be large
  - Ill-conditioning, high contrasts, geometric singularities
  - Need preconditioners or **may not converge at all**
- Inefficient for multiple right-hand sides
  - Time-stepping, inverse problems, optimization, design

Direct vs. iterative solvers

- Direct solvers: no iteration (e.g., Gaussian elimination)
- Why direct solvers? Compare with iterative methods

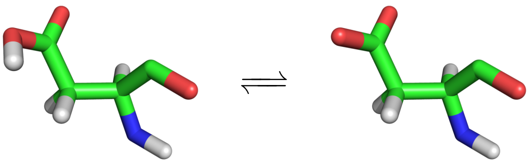
Iterative solvers

- GMRES, CG, relaxation methods, multigrid, etc.
- Can achieve linear complexity under certain conditions
- But number of iterations can be large
  - Ill-conditioning, high contrasts, geometric singularities
  - Need preconditioners or **may not converge at all**
- Inefficient for multiple right-hand sides
  - Time-stepping, inverse problems, optimization, design

Direct solvers

- No convergence issues, much more robust
- Typically very fast solves following initial factorization
- However, classical direct methods can be extremely expensive

Application: protein pK_a calculations

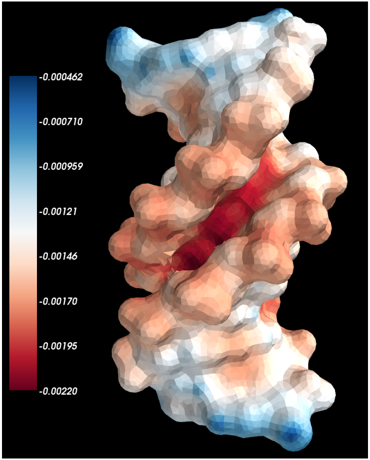


$$\mathrm{p}K_\mathrm{a} \equiv -\log_{10}\frac{[\mathrm{A}]\,[\mathrm{H}]}{[\mathrm{AH}]} = \frac{\beta}{\ln 10}\Delta G^{\mathrm{p}}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}}$$

$$\begin{aligned}\Delta G^{\mathrm{p}}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}} &= \Delta G^{\mathrm{s}}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}} + \Delta G^{\mathrm{s}\to\mathrm{p}}_{\mathrm{A}} - \Delta G^{\mathrm{s}\to\mathrm{p}}_{\mathrm{AH}} \\ &= \underbrace{\Delta G^{\mathrm{s}}_{\mathrm{AH}\to\mathrm{A}+\mathrm{H}}}_{\text{experiment}} + \underbrace{\Delta G^{\mathrm{s}}_{\mathrm{A}\to\mathrm{AH}} - \Delta G^{\mathrm{p}}_{\mathrm{A}\to\mathrm{AH}}}_{\text{electrostatic only}}\end{aligned}$$

- Ionization behavior is important for enzymatic and structural properties

Application: protein pK_a calculations

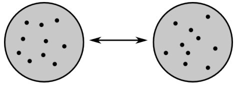


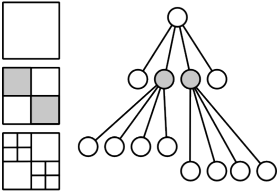
- Linearized Poisson-Boltzmann equation
- Discretize: $A(\Sigma)u = f(q)$
  - Σ: molecular geometry
  - q: atomic partial charges
- One matrix, $\gtrsim 100$ right-hand sides
- One solve for each of N_{titr} titrating sites
- If including comformational flexibility, then require $\mathcal{O}(N_{\text{titr}} N_{\text{rot}})$ solves

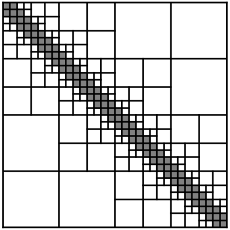
Potential for massive acceleration using fast direct solvers.

Previous fast direct solvers

- HSS matrices/recursive skeletonization/multifrontal
  - $\mathcal{O}(N)$ in 1D, $\mathcal{O}(N^{3/2})$ in 2D, $\mathcal{O}(N^2)$ in 3D
- $\mathcal{H}$-matrices: $\mathcal{O}(N \log^\alpha N)$ but with a **large** constant
- HSS/RS/MF with structured matrix algebra
  - $\mathcal{O}(N)$ in 2D, $\mathcal{O}(N^{4/3})$ in 3D
- All based on FMM-type hierarchical low-rank compression







Overview

Hierarchical interpolative factorization

- RS/MF + recursive dimensional reduction
- Same idea as using structured algebra but much simpler
- New matrix sparsification framework, generalized LU decomposition
- Linear or nearly linear complexity, small constants
- Works for IEs and PDEs in 2D and 3D
- Handles adaptivity and complex geometry

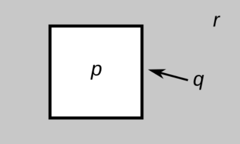
Tools: block elimination, interpolative decomposition, skeletonization

Focus on IEs in this talk; consider PDEs as a special case.

Block 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

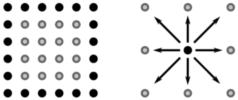
Interpolative decomposition

If $A_{:,q}$ 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

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



- Essentially a pivoted QR written slightly differently:

$$A_{:,(\hat{q},\check{q})} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} \approx Q_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}$$

$$\implies A_{:,\check{q}} \approx Q_1 R_{12} = \underbrace{Q_1 R_{11}}_{A_{:,\hat{q}}} \underbrace{\left(R_{11}^{-1} R_{12}\right)}_{T_q}$$

Interactions between separated regions are low-rank.

Skeletonization

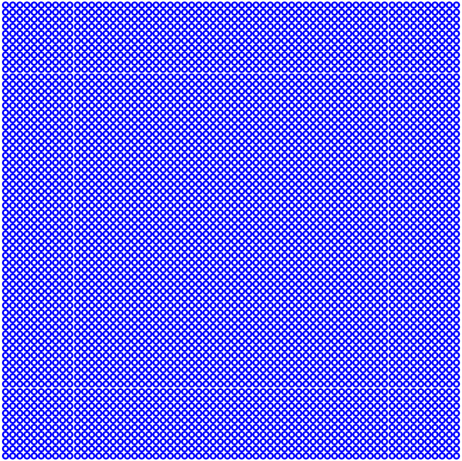
- Efficient elimination of redundant DOFs from dense matrices
- Let $A = \begin{bmatrix} A_{pp} & A_{pq} \\ A_{qp} & A_{qq} \end{bmatrix}$ with A_{pq} and A_{qp} low-rank
- Apply ID to $\begin{bmatrix} A_{qp} \\ A_{pq}^* \end{bmatrix}$: $\quad \begin{bmatrix} A_{q\check{p}} \\ A_{\check{p}q}^* \end{bmatrix} \approx \begin{bmatrix} A_{q\hat{p}} \\ A_{\hat{p}q}^* \end{bmatrix} T_p \implies \begin{matrix} A_{q\check{p}} \approx A_{q\hat{p}} T_p \\ A_{\check{p}q} \approx T_p^* A_{\hat{p}q} \end{matrix}$
- 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: $Q_p^* A Q_p \approx \begin{bmatrix} * & * & \\ * & A_{\hat{p}\hat{p}} & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix}$
- Block eliminate: $R_p^* Q_p^* A Q_p S_p \approx \begin{bmatrix} * & & \\ & * & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix}$

Algorithm: recursive skeletonization factorization

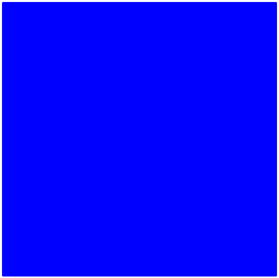
Build quadtree/octree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
end for

- Old algorithm in new factorization form
- Successive elimination of DOFs

RSF in 2D: level 0

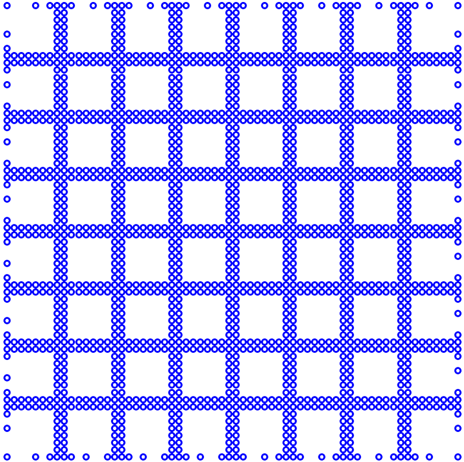


domain

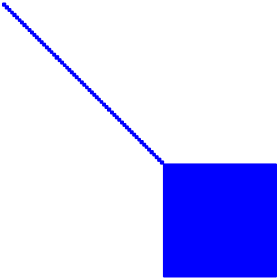


matrix

RSF in 2D: level 1

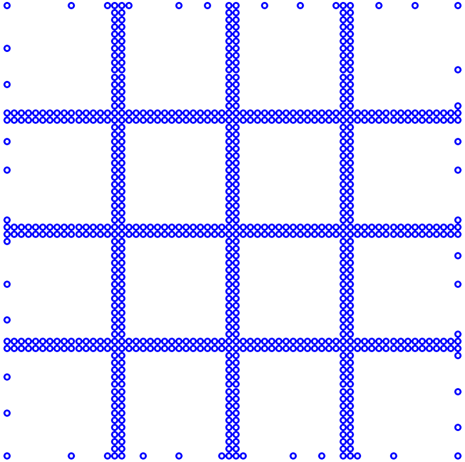


domain

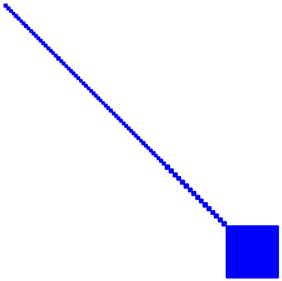


matrix

RSF in 2D: level 2

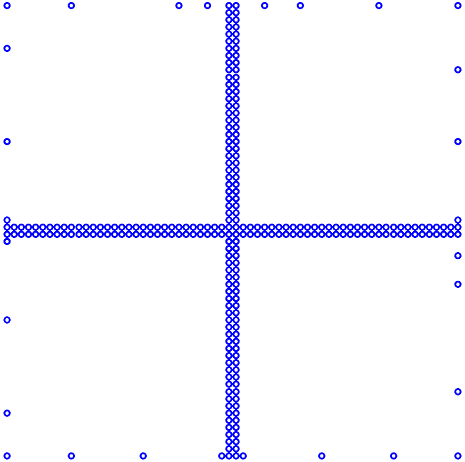


domain

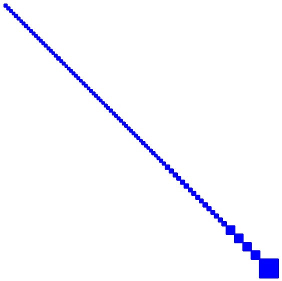


matrix

RSF in 2D: level 3

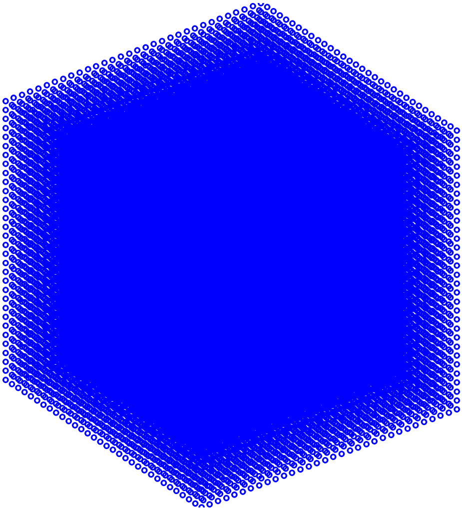


domain



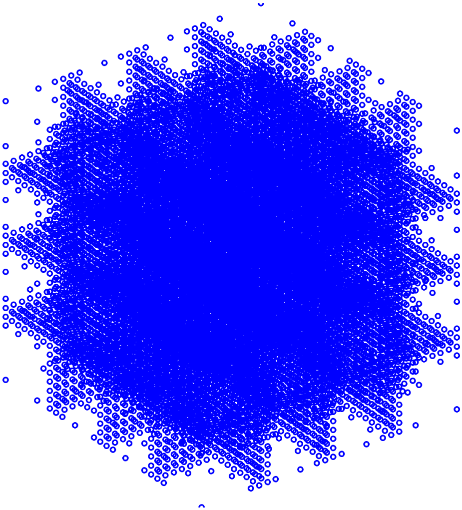
matrix

RSF in 3D: level 0

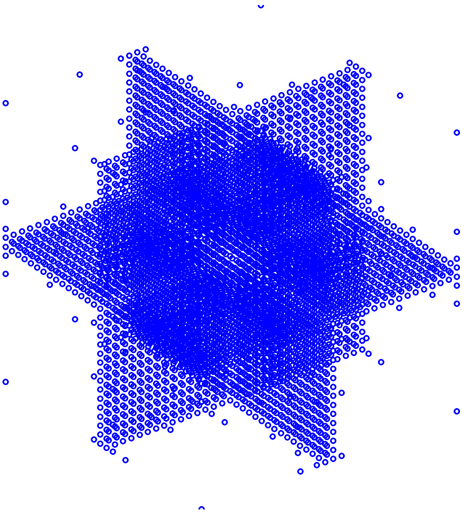


domain

RSF in 3D: level 1



RSF in 3D: level 2



domain

RSF analysis

- Skeletonization operators:

$$U_\ell = \prod_{c \in C_\ell} Q_c R_c, \quad V_\ell = \prod_{c \in C_\ell} Q_c S_c$$

$$Q_c = \begin{bmatrix} I & & \\ * & I & \\ & & I \end{bmatrix}, \quad R_c, S_c = \begin{bmatrix} I & * & \\ & I & \\ & & I \end{bmatrix}$$

- 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

RSF analysis

The cost is determined by the skeleton size.

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

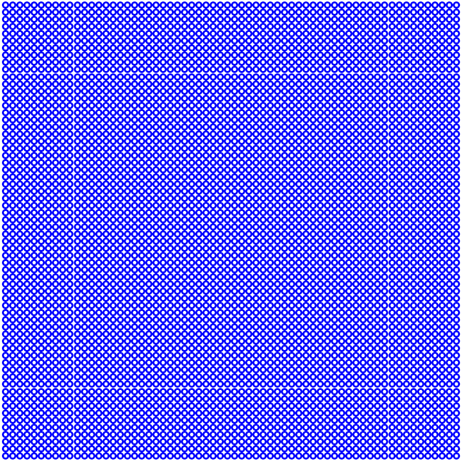
Question: How to reduce the skeleton size in 2D and 3D?

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

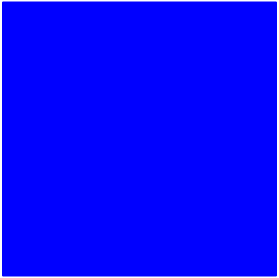
Algorithm: hierarchical interpolative factorization for IEs in 2D

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

HIF-IE in 2D: level 0

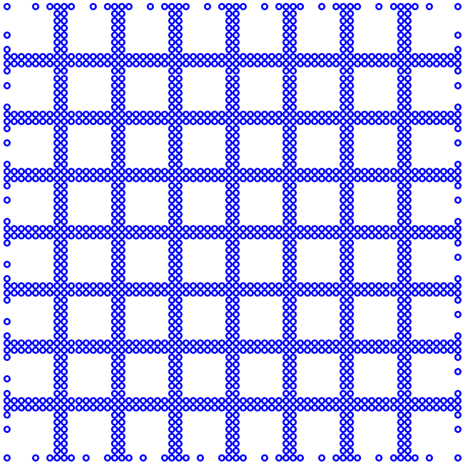


domain

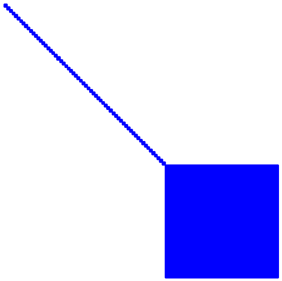


matrix

HIF-IE in 2D: level 1/2

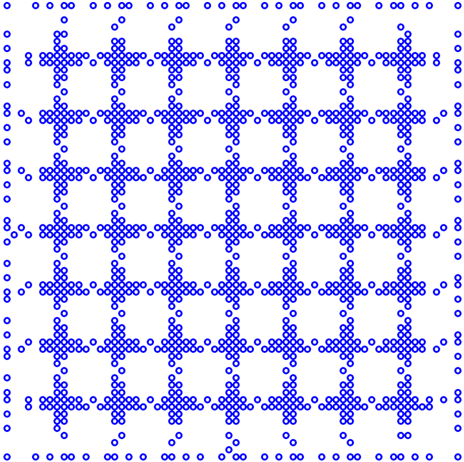


domain

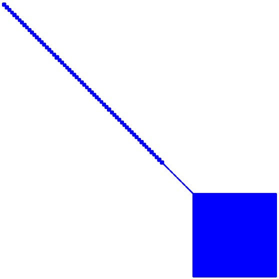


matrix

HIF-IE in 2D: level 1

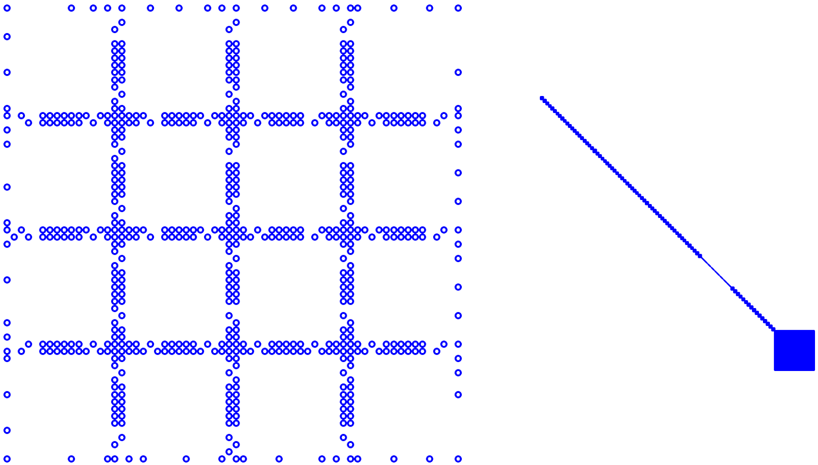


domain



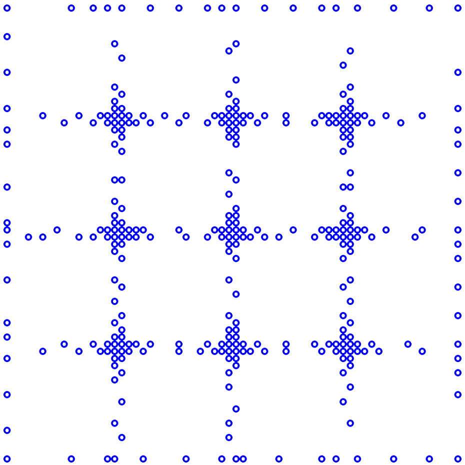
matrix

HIF-IE in 2D: level 3/2

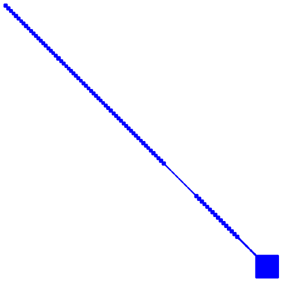


domain matrix

HIF-IE in 2D: level 2

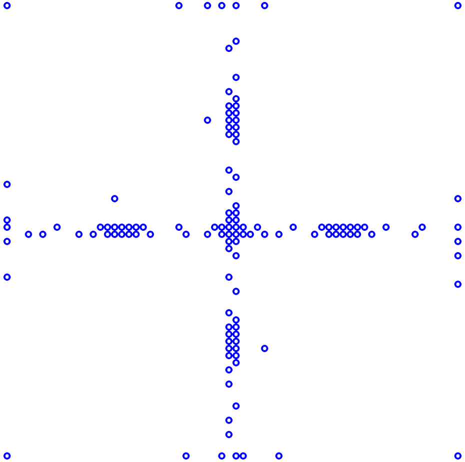


domain

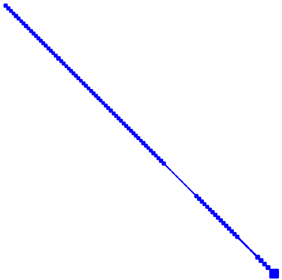


matrix

HIF-IE in 2D: level 5/2

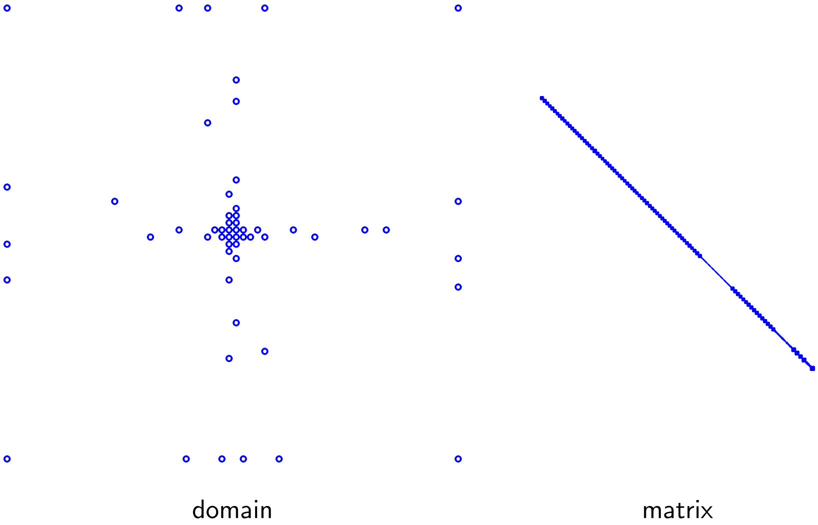


domain

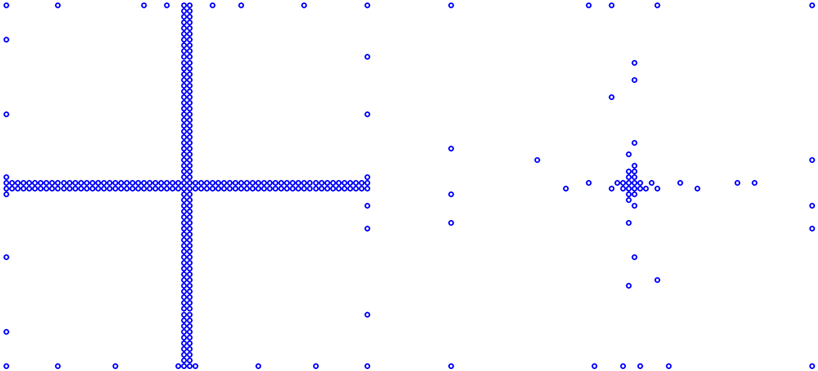


matrix

HIF-IE in 2D: level 3

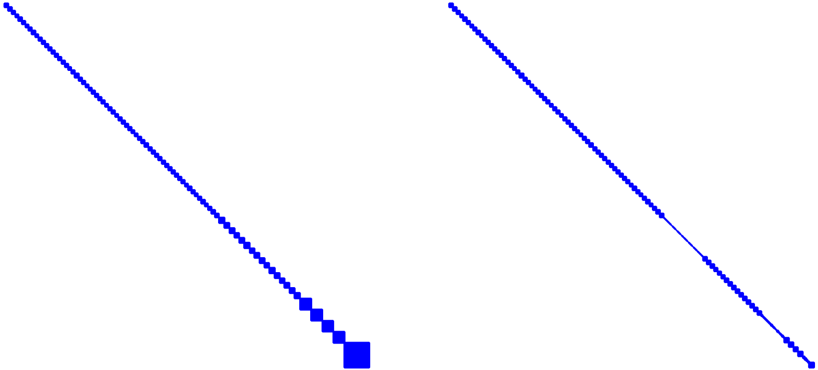


RSF vs. HIF-IE in 2D



RSF HIF-IE

RSF vs. HIF-IE in 2D

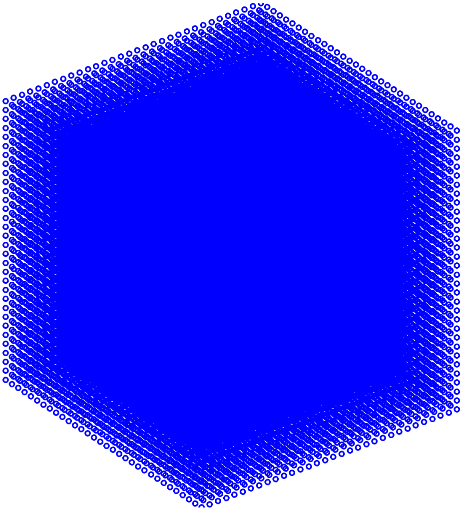


RSF HIF-IE

Algorithm: hierarchical interpolative factorization for IEs in 3D

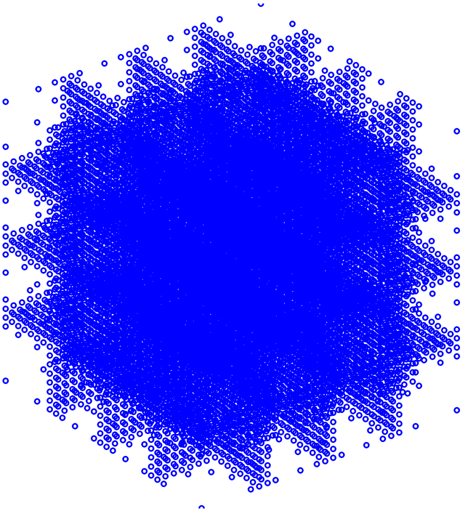
Build octree.
for each level $\ell = 0, 1, 2, \ldots, L$ from finest to coarsest **do**
    Let C_ℓ be the set of all cells on level ℓ.
    for each cell $c \in C_\ell$ **do**
        Skeletonize remaining DOFs in c.
    end for
    Let $C_{\ell+1/3}$ be the set of all faces on level ℓ.
    for each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

HIF-IE in 3D: level 0



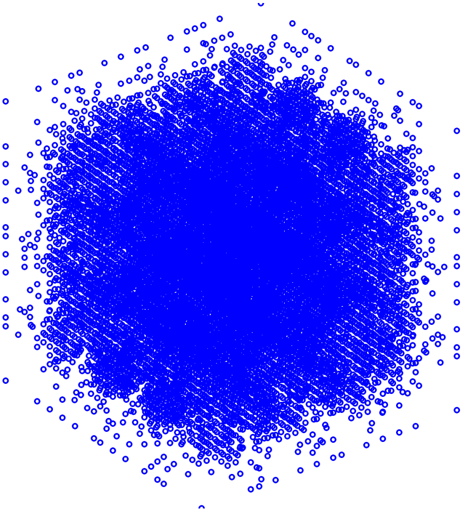
domain

HIF-IE in 3D: level 1/3



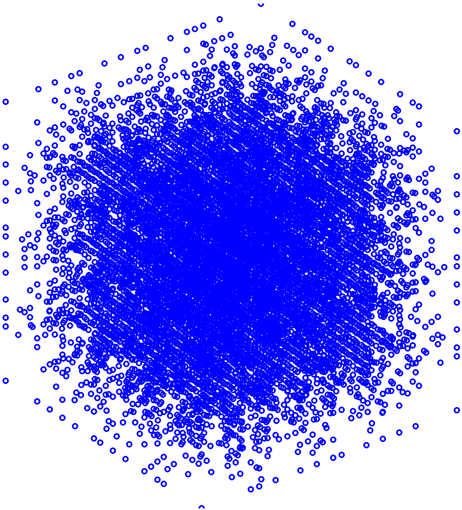
domain

HIF-IE in 3D: level 2/3



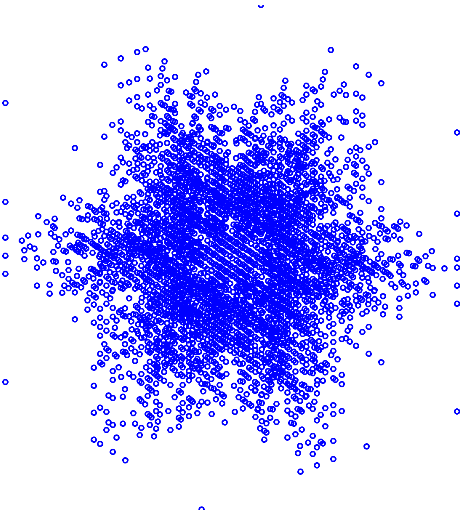
domain

HIF-IE in 3D: level 1



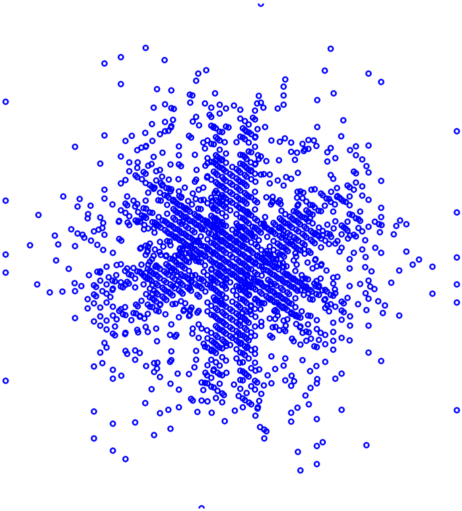
domain

HIF-IE in 3D: level 4/3



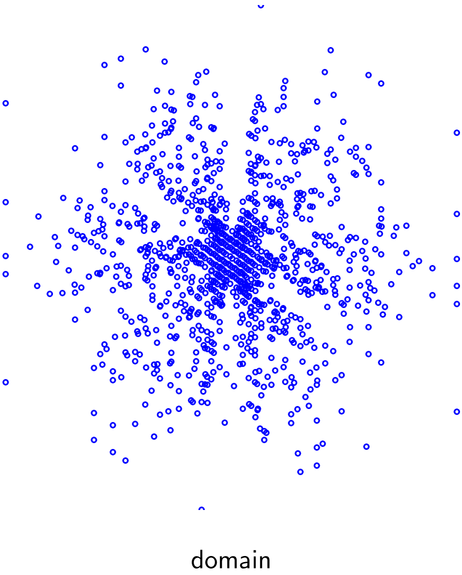
domain

HIF-IE in 3D: level 5/3

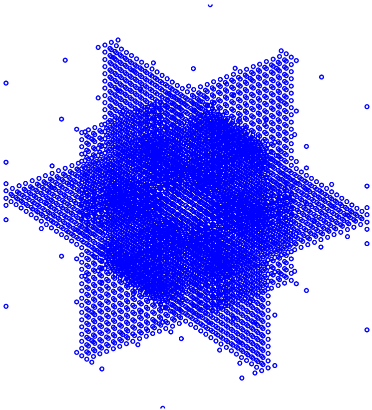


domain

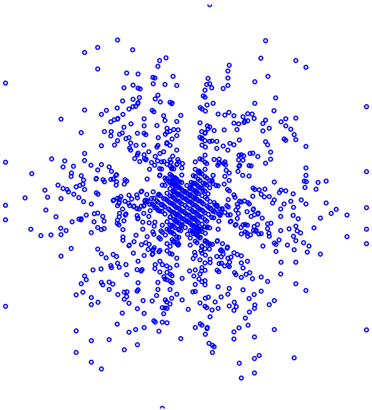
HIF-IE in 3D: level 2



RSF vs. HIF-IE in 3D



RSF



HIF-IE

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

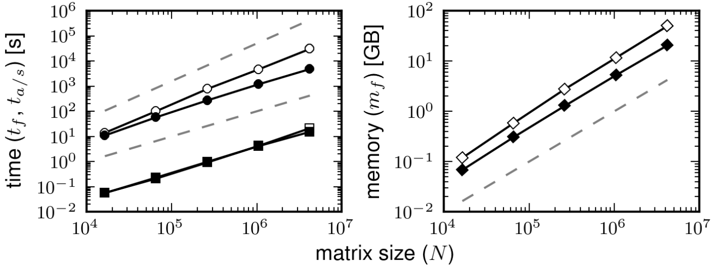
Conjecture:

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

Numerical results in 2D

First-kind volume IE on the unit square with

$$a(x) \equiv 0, \quad K(x, y) = -\frac{1}{2\pi} \log \|x - y\|.$$

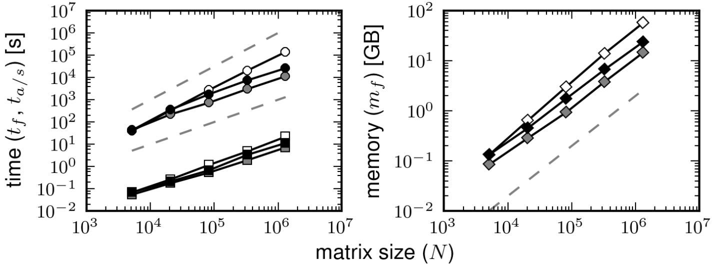


- **rskelf2** (white), **hifie2** (black)
- Factorization time ($\circ$), solve time ($\square$), memory ($\diamond$)
- Precision $\epsilon = 10^{-6}$

Numerical results in 3D

Second-kind boundary IE on the unit sphere with

$$a(x) \equiv -\frac{1}{2}, \quad K(x,y) = \frac{\partial}{\partial \nu(y)} \frac{1}{4\pi \|x-y\|}.$$

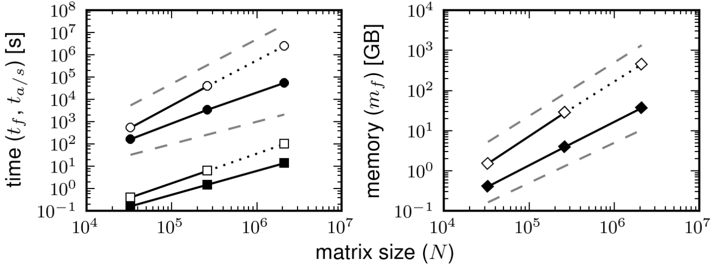


- **rskelf3** (white), **hifie3** (gray), **hifie3x** (black)
- Factorization time (○), solve time (□), memory (◇)
- Precision $\epsilon = 10^{-3}$

Numerical results in 3D

First-kind volume IE on the unit cube with

$$a(x) \equiv 0, \quad K(x, y) = \frac{1}{4\pi\|x - y\|}.$$



- **rskelf3** (white), **hifie3** (black)
- Factorization time ($\circ$), solve time ($\square$), memory ($\diamond$)
- Precision $\epsilon = 10^{-3}$

Conclusions

- Efficient factorization of integral operators in 2D and 3D
  - Fast matrix-vector multiplication
  - Fast direct solver at high accuracy, preconditioner otherwise
  - Empirical **linear complexity** but no proof yet
- Sparsification and elimination (skeletonization) via the ID
- Dimensional reduction by alternating between cells, faces, and edges
- Analogous techniques for differential operators; optimize by exploiting sparsity
- **Extensions**: general structured matrices, $A^{1/2}$, $\log\det A$, $\operatorname{diag} A^{-1}$

- *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?

MATLAB codes available at `https://github.com/klho/FLAM/`.