

Hierarchical interpolative factorization

Kenneth L. Ho (Stanford)

Joint work with Lexing Ying

Applied Math Seminar, Colorado State University, Mar 2014

Introduction

Elliptic PDEs in integral or differential form:

$$\begin{aligned} a(x)u(x) + \int_{\Omega} K(x, y)u(y) d\Omega(y) &= f(x) \\ -\nabla \cdot (a(x)\nabla u(x)) + v(x)u(x) &= f(x) \end{aligned}$$

- ▶ Fundamental to physics and engineering
- ▶ Interested in 2D/3D, complex geometry
- ▶ Discretize → structured linear system $Ax = b$

Goal: fast and accurate algorithms for the discrete operator

- ▶ Fast **direct** solver, good preconditioner
- ▶ 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

Iterative solvers

- ▶ GMRES, CG, relaxation methods, multigrid, etc.
- ▶ Can achieve linear complexity under certain conditions
 - For IEs, accelerate with FMM, treecode, \mathcal{H} -matrices
- ▶ 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, more robust
- ▶ Typically very fast solves following an initial factorization
- ▶ However, classical direct methods can be expensive
 - Our aim is to accelerate to linear complexity

Previous work: direct solvers for IEs

HSS matrices/recursive skeletonization

- ▶ $\mathcal{O}(N)$ in 1D, $\mathcal{O}(N^{3/2})$ in 2D, $\mathcal{O}(N^2)$ in 3D
- ▶ Precursors: Greengard, Rokhlin (1991); Chen (2002)
- ▶ HSS: Chandrasekaran, Gu, Pals (2006);
Chandrasekaran, Dewilde, Gu, Lyons, Pals (2006)
- ▶ RS: Martinsson, Rokhlin (2005); Gillman, Young, Martinsson (2012);
Ho, Greengard (2012)
- ▶ Related: Kong, Bremer, Rokhlin (2011); Ambikasaran, Darve (2013)

$\mathcal{H}/\mathcal{H}^2$ -matrices

- ▶ $\mathcal{O}(N)$ or $\mathcal{O}(N \log^\alpha N)$, but with a **large** constant
- ▶ Hackbusch (1999); Hackbusch, Khoromskij (2000); Hackbusch, Börm (2002)

HSS/RS with structured matrix algebra: Corona, Martinsson, Zorin (2013)

- ▶ $\mathcal{O}(N)$ in 2D, but complex geometry can be difficult

Previous work: direct solvers for PDEs

Nested dissection/multifrontal: George (1973); Duff, Reid (1983)

- ▶ $\mathcal{O}(N^{3/2})$ in 2D, $\mathcal{O}(N^2)$ in 3D, with **very** small constants

$\mathcal{H}/\mathcal{H}^2$ -matrices

- ▶ $\mathcal{O}(N)$ or $\mathcal{O}(N \log^\alpha N)$, but with a large constant
- ▶ Hackbusch et al.; Bebendorf, Hackbusch (2003); Le Borne, Grasedyck (2006); Börm (2010)

MF with \mathcal{H} -matrix algebra

- ▶ **$\mathcal{O}(N)$** , improved constant
- ▶ Grasedyck, Kriemann, Le Borne (2009); Schmitz, Ying (2012, 2014)

MF with HSS algebra

- ▶ **$\mathcal{O}(N)$ in 2D, $\mathcal{O}(N^{4/3})$ in 3D**
- ▶ Xia, Chandrasekaran, Gu, Li (2009); Xia, Gu (2010); Li, Gu, Wu, Xia (2012); Gillman, Martinsson (2013)
- ▶ Related: Martinsson (2009)

Overview

Hierarchical interpolative factorization

- ▶ RS/MF + recursive dimensional reduction
- ▶ Works for both integral and differential equations
- ▶ Linear or nearly linear complexity, small constants
- ▶ Same idea as with using structured algebra but in a new matrix framework
- ▶ Explicit **sparsification**, generalized LU decomposition
- ▶ Extends to 3D, complex geometry, etc.

Tools: block elimination, interpolative decomposition, skeletonization

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

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

such that

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

- ▶ Essentially a pivoted QR written slightly differently
- ▶ Can be computed adaptively to any specified precision
- ▶ Fast randomized algorithms are available

Interactions between separated regions are low-rank.

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 $\begin{bmatrix} A_{qp} \\ A_{pq}^* \end{bmatrix}$: $\begin{bmatrix} A_{q\check{p}} \\ A_{\check{p}q}^* \end{bmatrix} \approx \begin{bmatrix} A_{q\hat{p}} \\ A_{\hat{p}q}^* \end{bmatrix} T_p \implies \begin{aligned} A_{q\check{p}} &\approx A_{q\hat{p}} T_p \\ A_{\check{p}q} &\approx T_p^* A_{\hat{p}q} \end{aligned}$
- ▶ 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: $R_p^* Q_p^* A Q_p S_p \approx \begin{bmatrix} * & & \\ & * & A_{\hat{p}q} \\ & A_{q\hat{p}} & A_{qq} \end{bmatrix}$

Integral equations

Algorithm: recursive skeletonization

Build quadtree/octree.

for each level $\ell = 0, 1, 2, \dots, L$ from finest to coarsest **do**

 Let C_ℓ be the set of all cells on level ℓ .

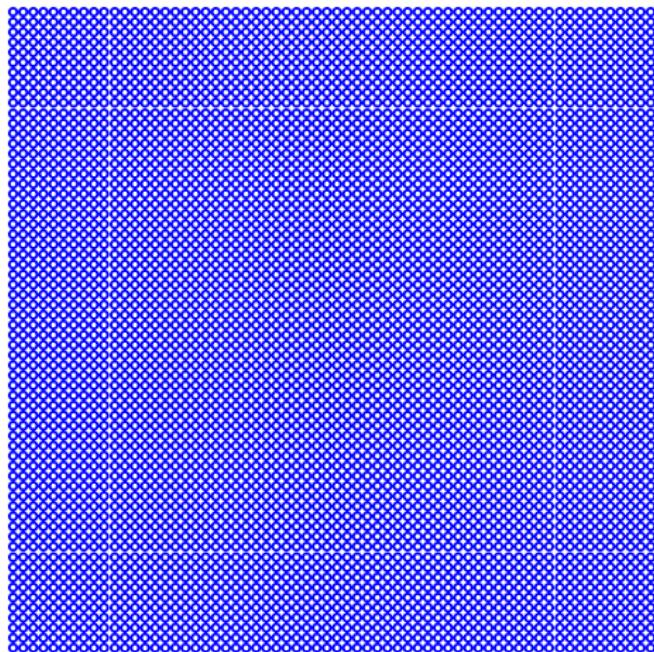
for each cell $c \in C_\ell$ **do**

 Skeletonize remaining DOFs in c .

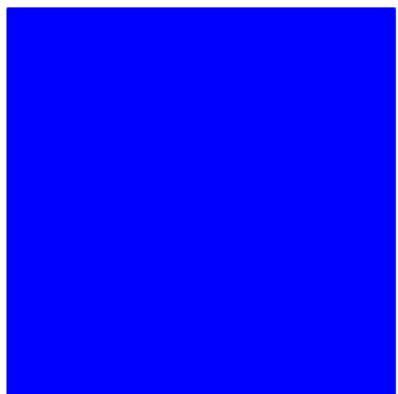
end for

end for

RS in 2D: level 0

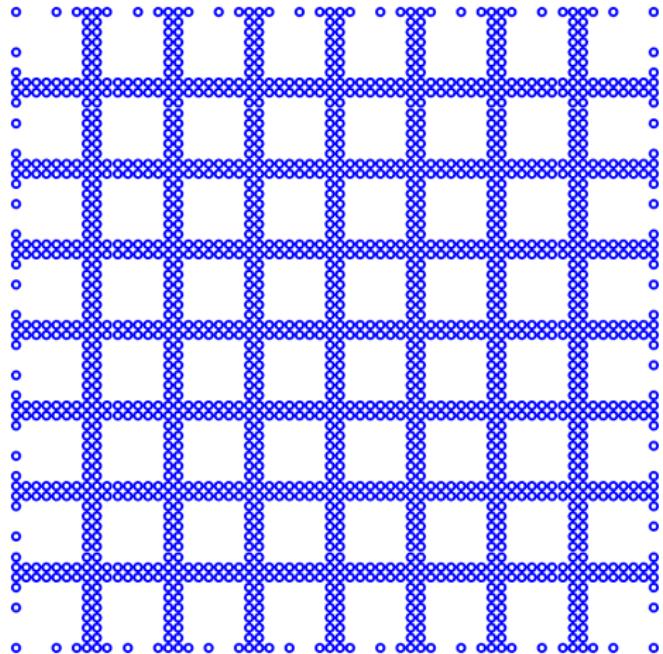


domain

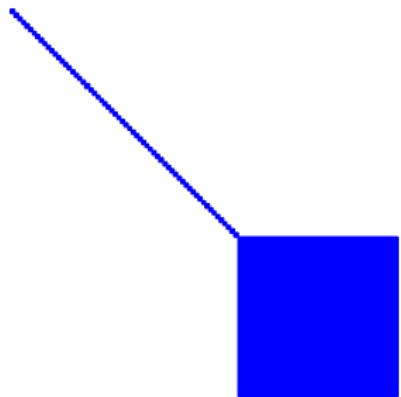


matrix

RS in 2D: level 1

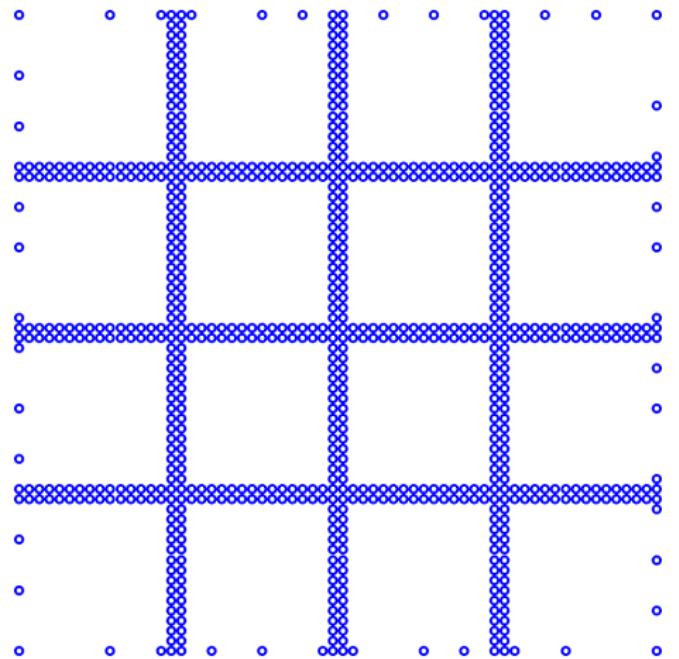


domain

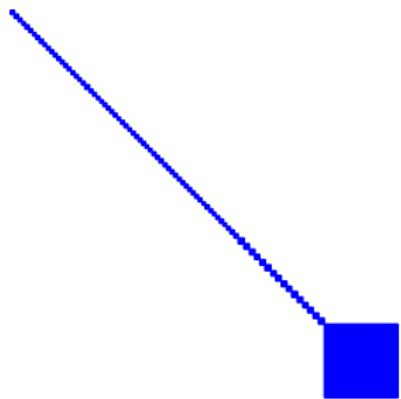


matrix

RS in 2D: level 2

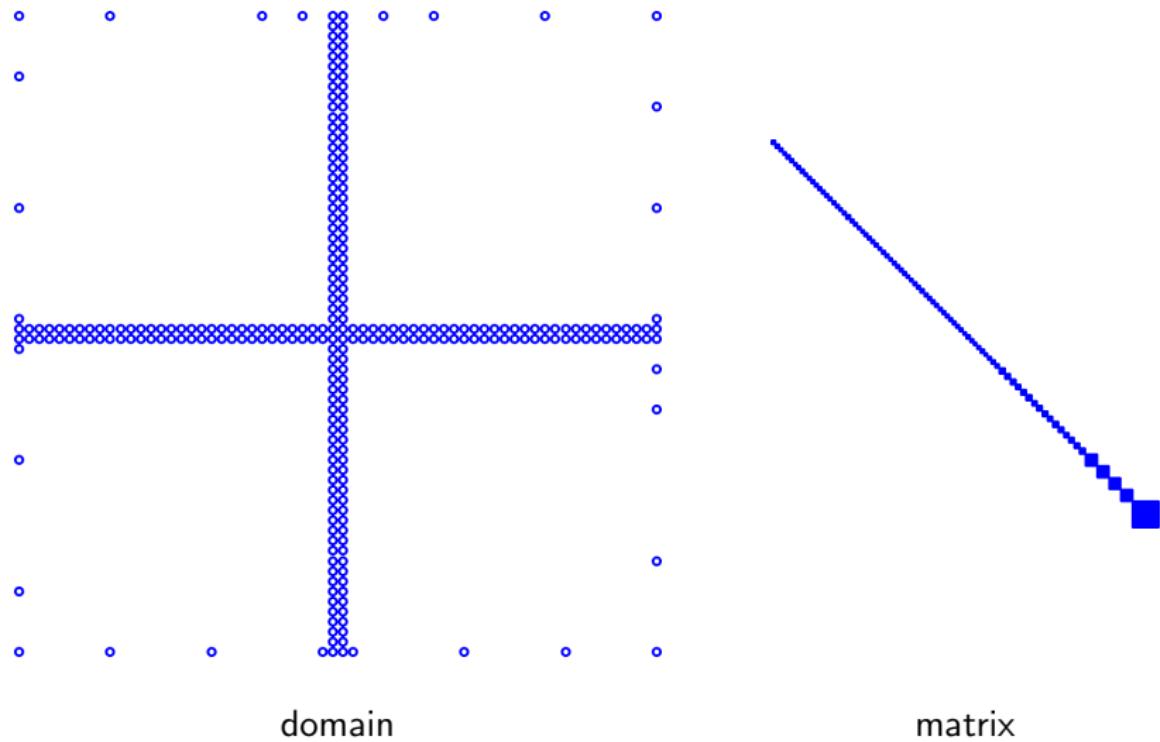


domain

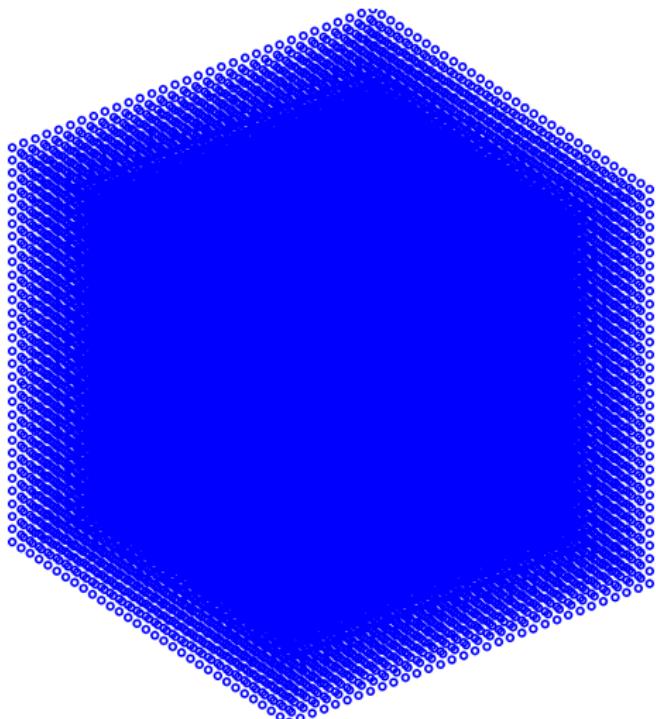


matrix

RS in 2D: level 3

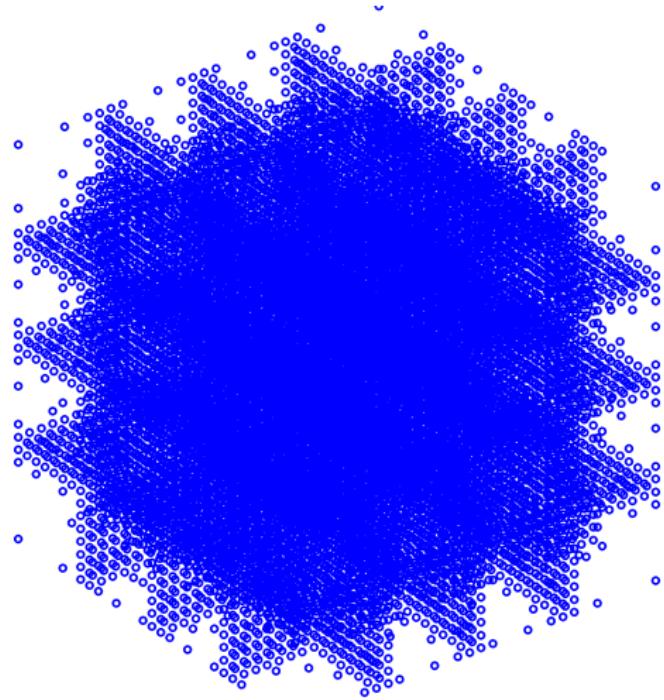


RS in 3D: level 0



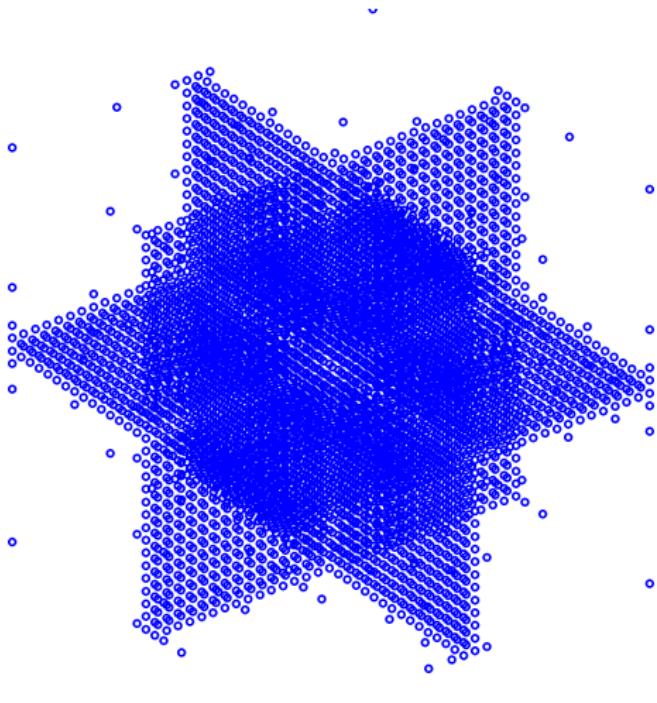
domain

RS in 3D: level 1



domain

RS in 3D: level 2



domain

RS analysis

- ▶ Skeletonization operators:

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

- ▶ Block diagonalization:

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

- ▶ Generalized LU decomposition:

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

- ▶ Fast direct **solver** or preconditioner

RS 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 $\ell = 0, 1, 2, \dots, L$ 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 .

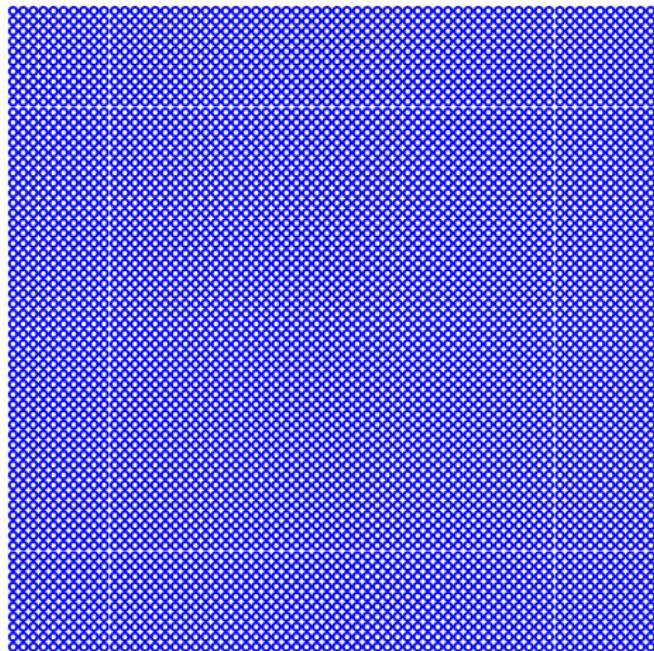
for each cell $c \in C_{\ell+1/2}$ **do**

 Skeletonize remaining DOFs in c .

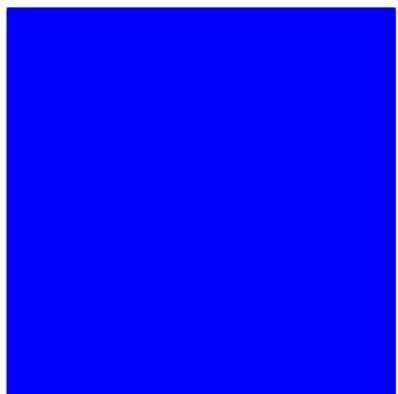
end for

end for

HIF-IE in 2D: level 0

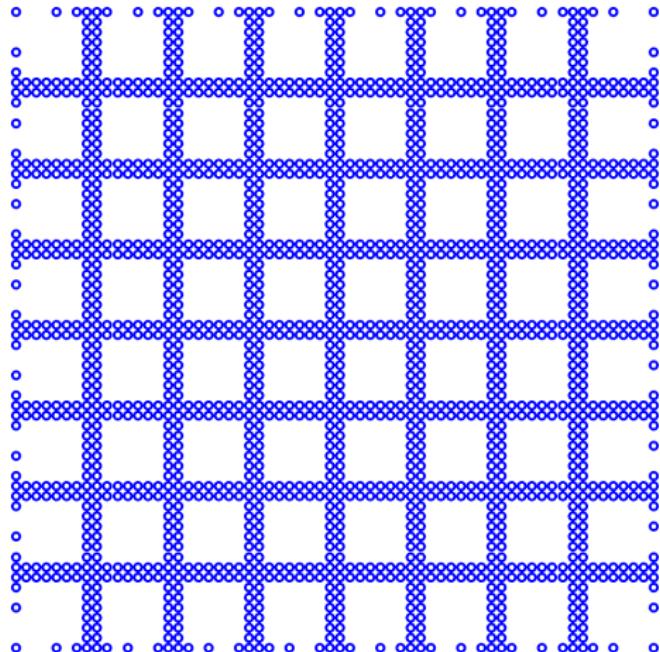


domain

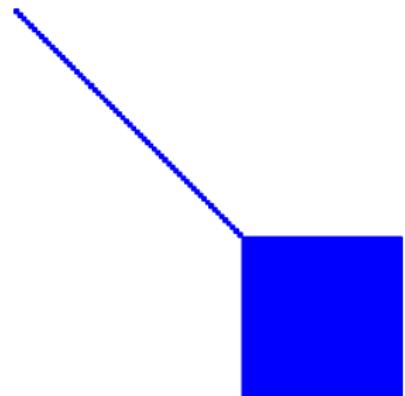


matrix

HIF-IE in 2D: level 1/2

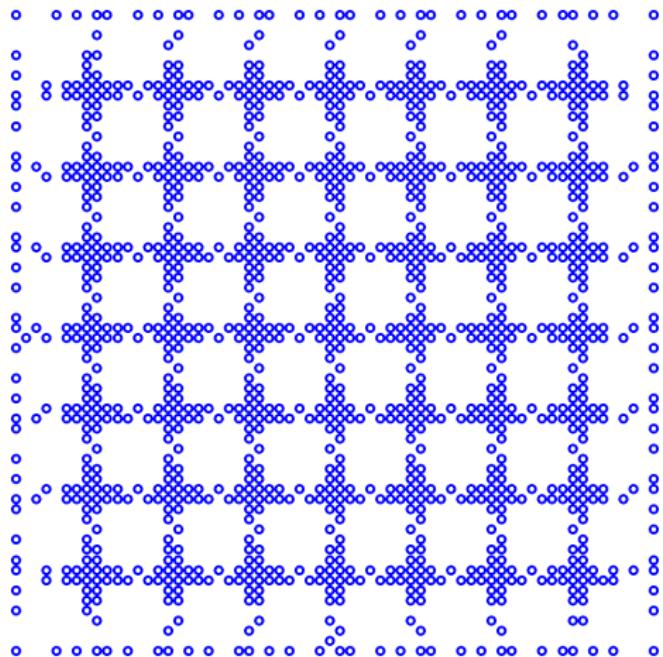


domain

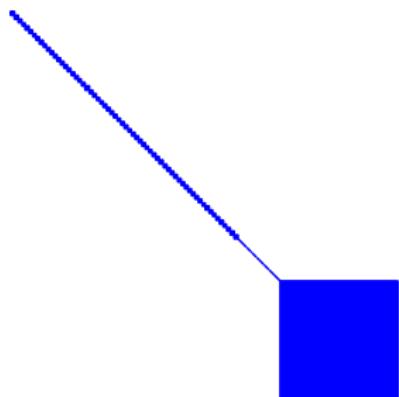


matrix

HIF-IE in 2D: level 1

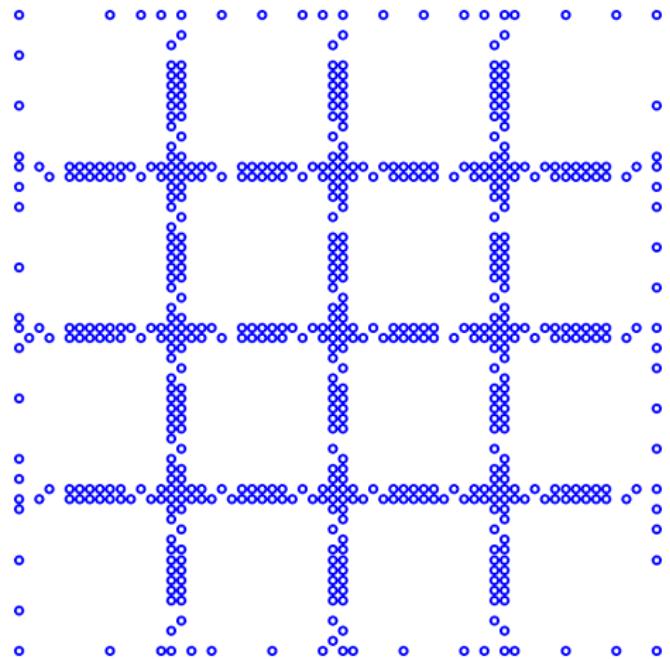


domain

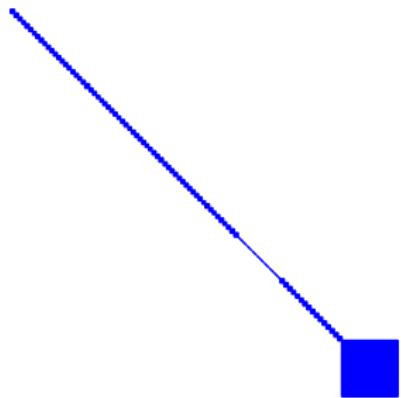


matrix

HIF-IE in 2D: level 3/2

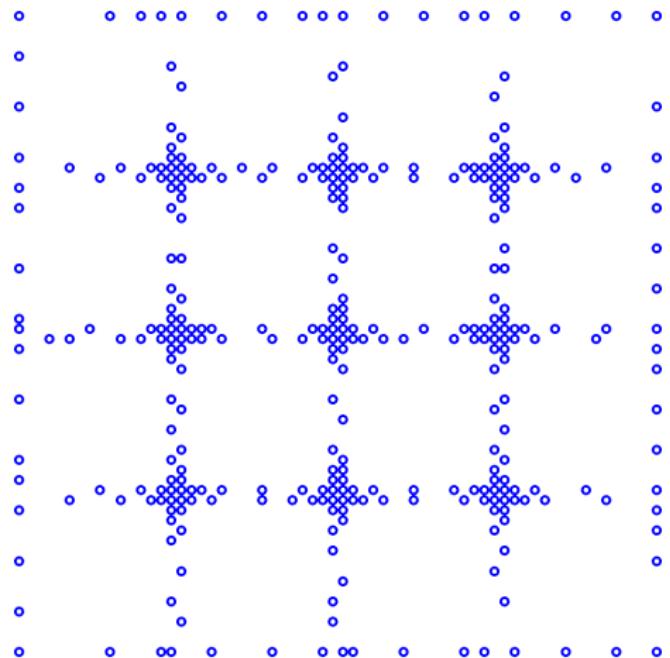


domain

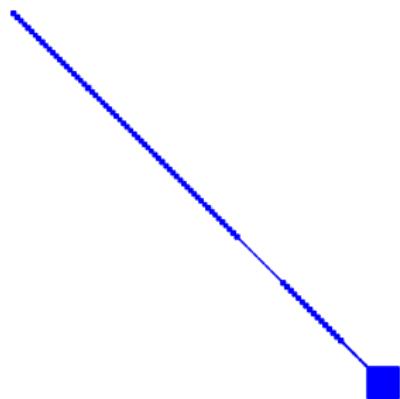


matrix

HIF-IE in 2D: level 2

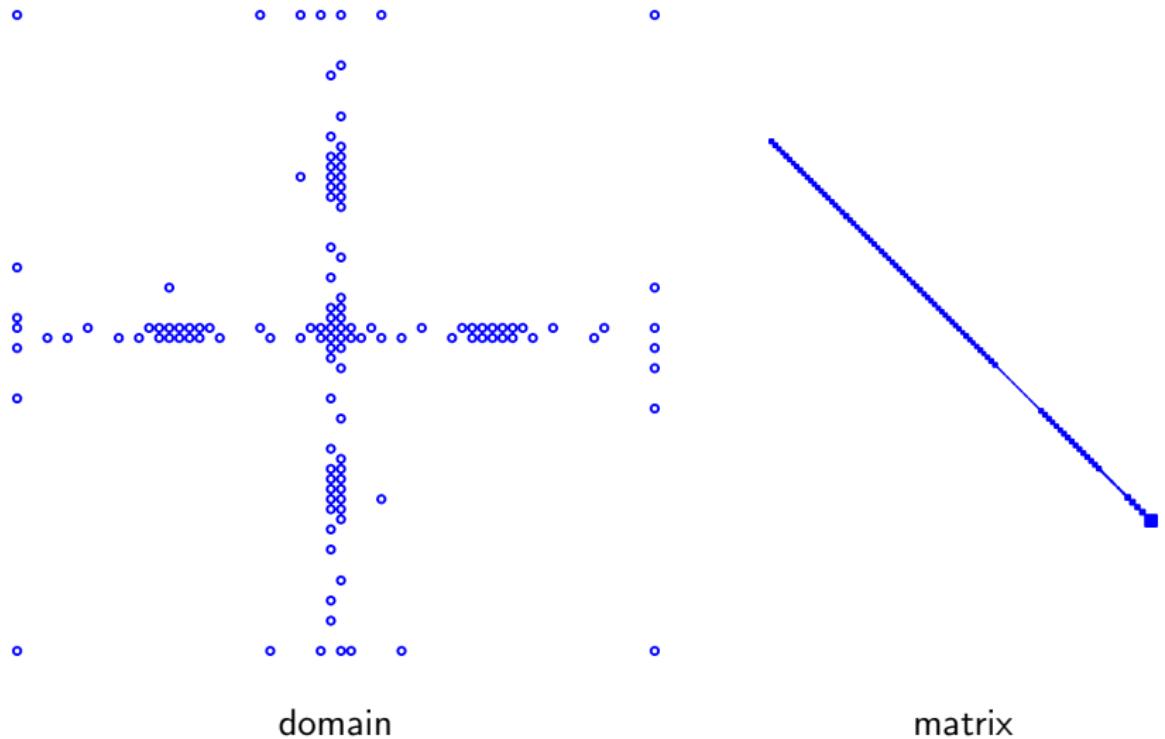


domain

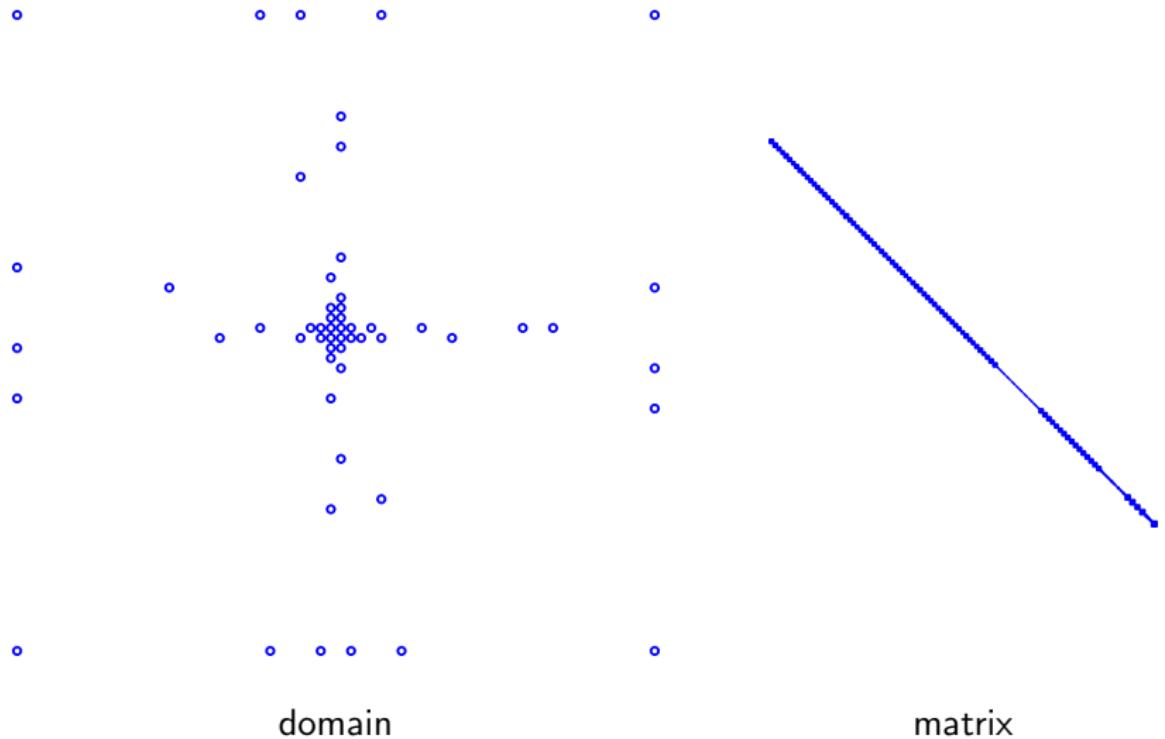


matrix

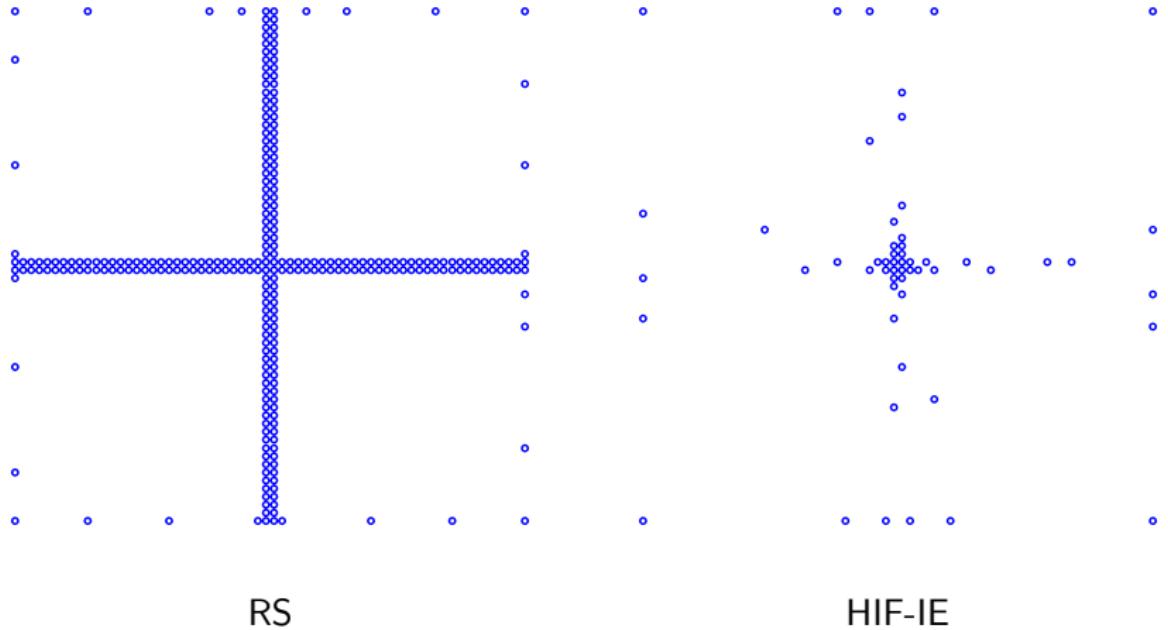
HIF-IE in 2D: level 5/2



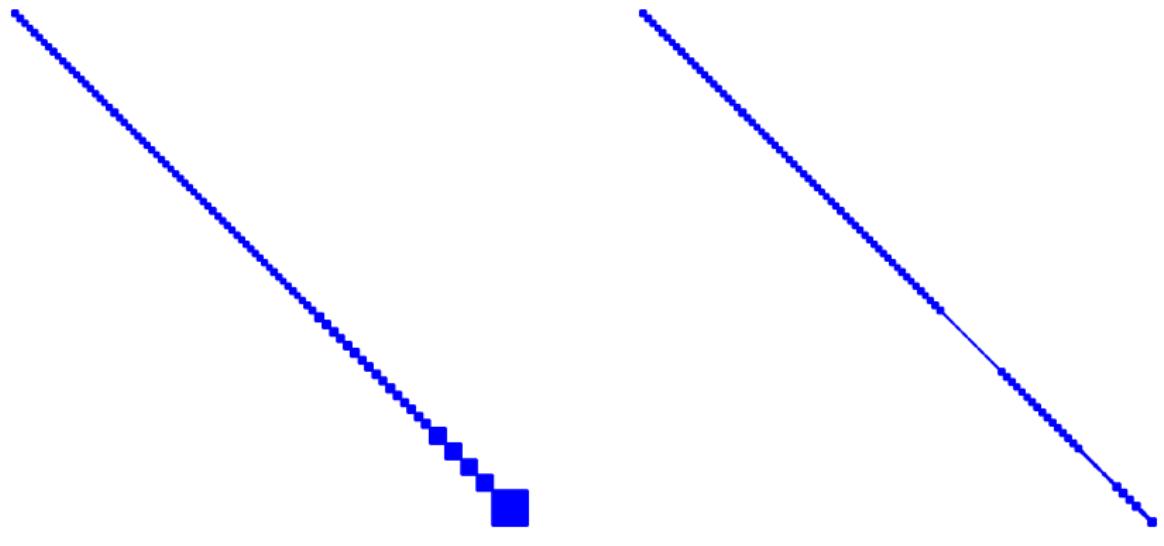
HIF-IE in 2D: level 3



RS vs. HIF-IE in 2D



RS vs. HIF-IE in 2D



RS

HIF-IE

Algorithm: hierarchical interpolative factorization for IEs in 3D

Build octree.

for each level $\ell = 0, 1, 2, \dots, L$ 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 .

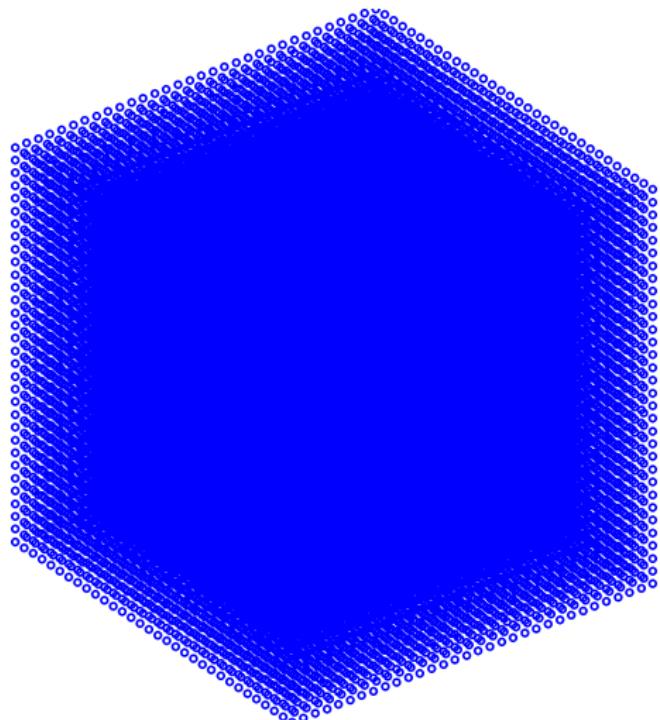
for each cell $c \in C_{\ell+2/3}$ **do**

 Skeletonize remaining DOFs in c .

end for

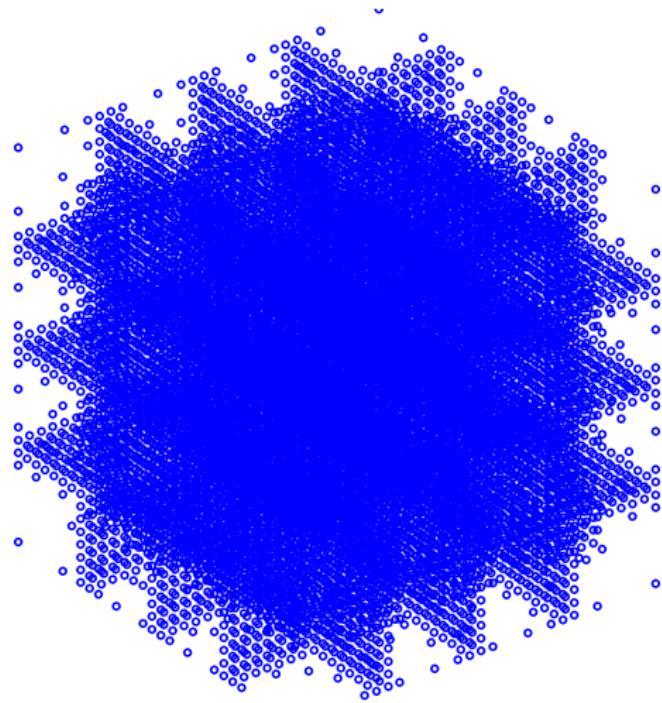
end for

HIF-IE in 3D: level 0



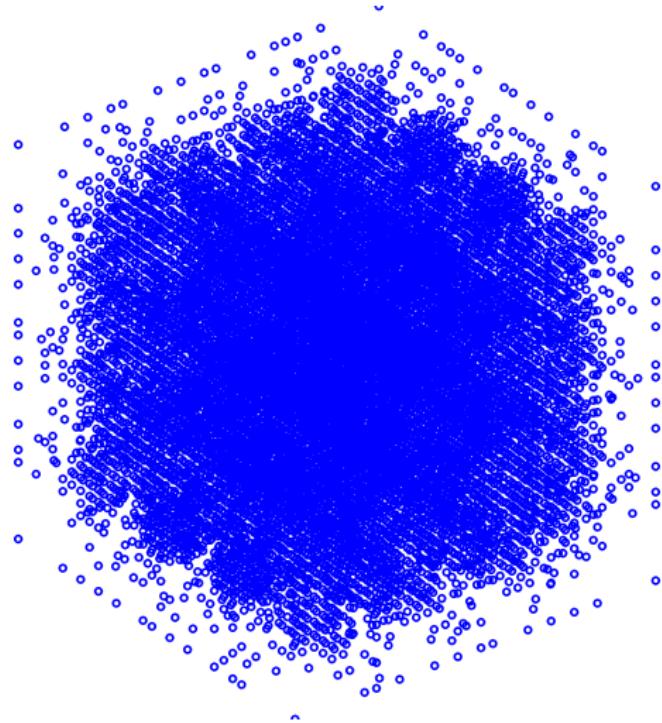
domain

HIF-IE in 3D: level 1/3



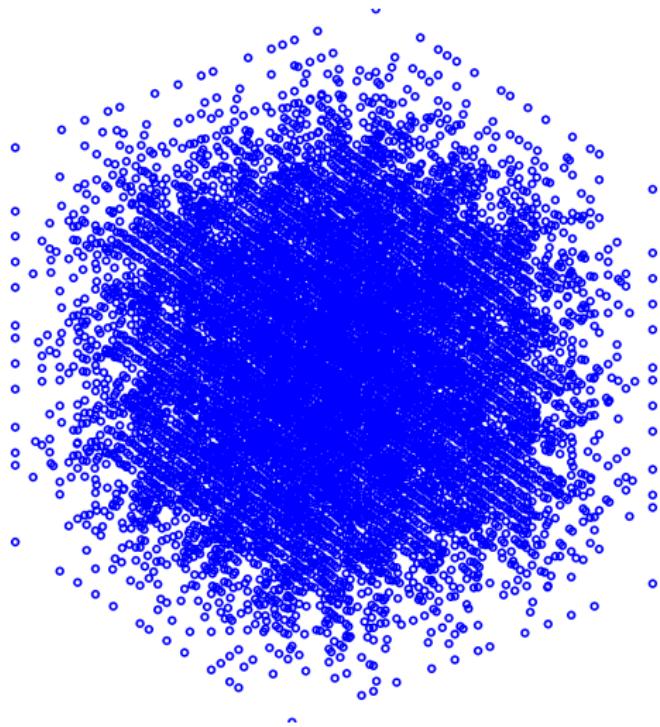
domain

HIF-IE in 3D: level 2/3



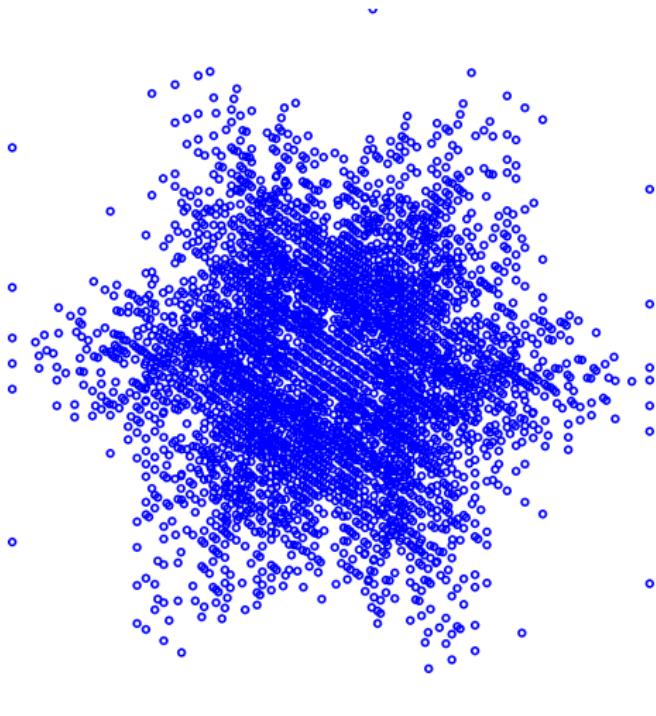
domain

HIF-IE in 3D: level 1



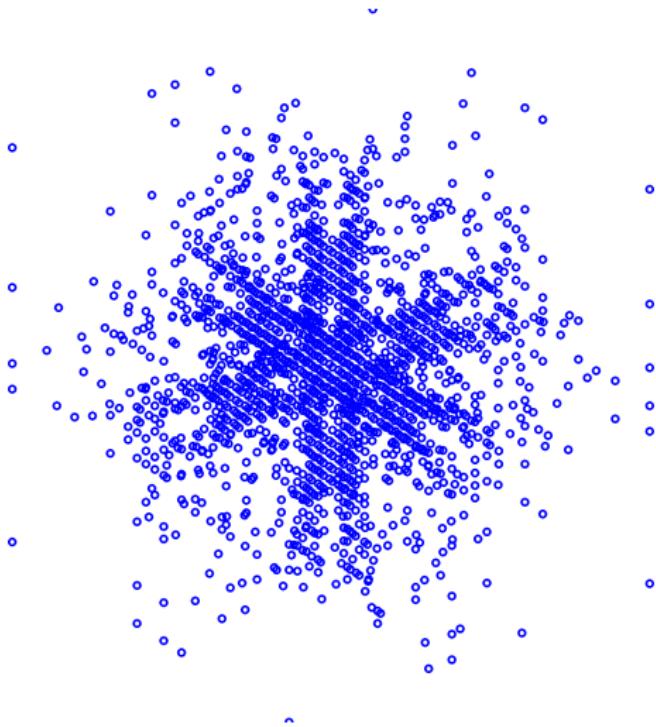
domain

HIF-IE in 3D: level 4/3



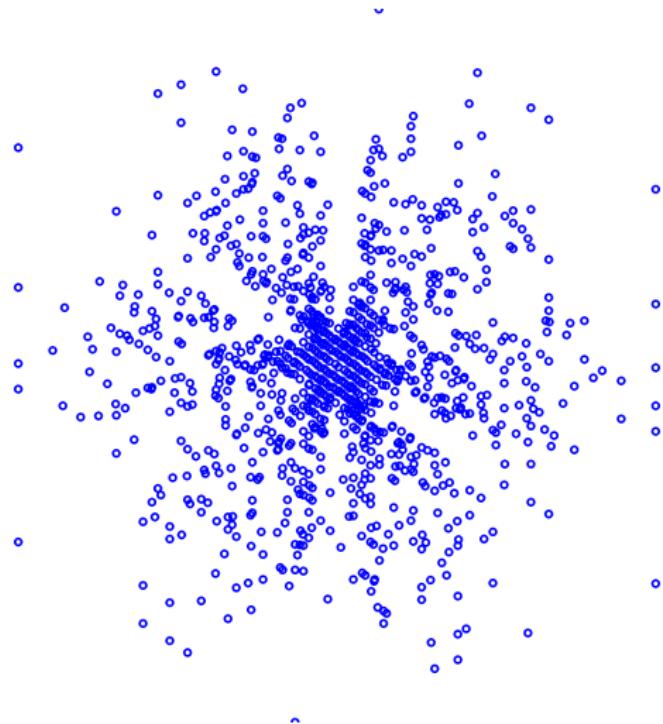
domain

HIF-IE in 3D: level 5/3



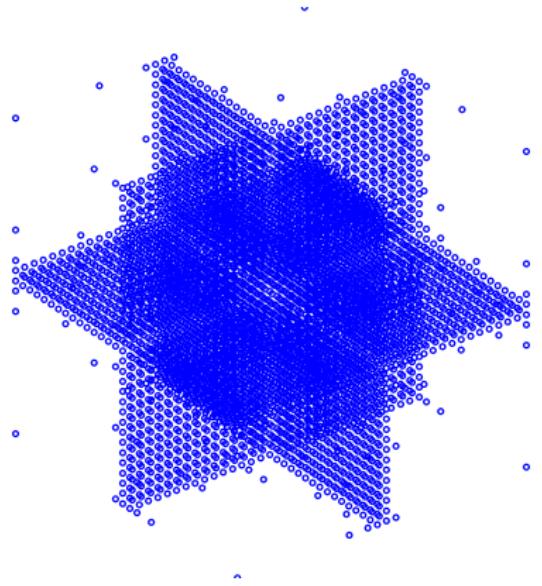
domain

HIF-IE in 3D: level 2

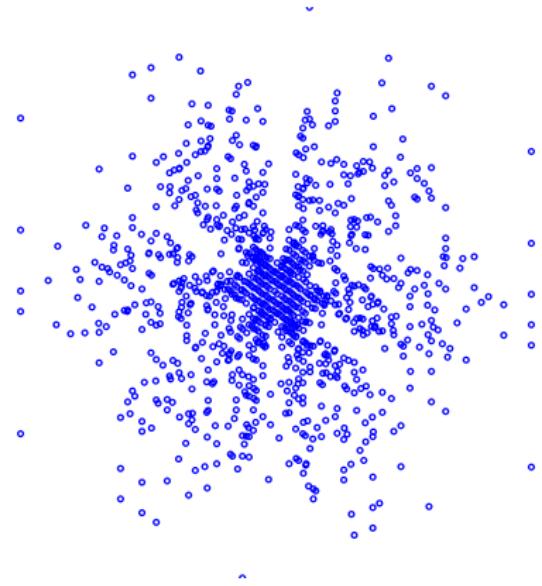


domain

RS vs. HIF-IE in 3D



RS



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** integral equation on a **square** with

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

ϵ	N	$ \hat{c} $	m_f (GB)	t_f (s)	$t_{a/s}$ (s)	e_a	e_s	n_i
10^{-3}	256^2	19	9.8e-2	1.0e+1	1.6e-1	1.8e-04	1.1e-2	8
	512^2	20	3.8e-1	4.3e+1	6.3e-1	1.6e-04	1.6e-2	8
	1024^2	20	1.5e+0	1.8e+2	2.6e+0	2.1e-04	1.4e-2	9
	2048^2	21	6.1e+0	7.5e+2	1.1e+1	2.2e-04	3.4e-2	9
10^{-6}	256^2	85	3.0e-1	2.7e+1	1.2e-1	2.0e-07	1.6e-5	3
	512^2	99	1.3e+0	1.3e+2	5.0e-1	1.3e-07	2.3e-5	3
	1024^2	115	5.4e+0	5.9e+2	2.1e+0	2.5e-07	3.4e-5	3
10^{-9}	256^2	132	4.4e-1	4.5e+1	1.2e-1	7.8e-11	1.3e-8	2
	512^2	155	1.8e+0	2.1e+2	4.9e-1	1.1e-10	1.6e-8	2
	1024^2	181	7.5e+0	9.7e+2	2.0e+0	1.8e-10	3.1e-8	2

Numerical results in 3D

Second-kind **boundary** integral equation on a **sphere** with

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

ϵ	N	$ \hat{c} $	m_f (GB)	t_f (s)	$t_{a/s}$ (s)	e_a	e_s
10^{-3}	20480	201	1.4e-1	9.8e+0	3.8e-2	7.2e-4	7.1e-4
	81920	307	5.6e-1	5.0e+1	1.8e-1	1.8e-3	1.8e-3
	327680	373	2.1e+0	2.2e+2	7.5e-1	3.8e-3	3.7e-3
	1310720	440	8.1e+0	8.9e+2	3.2e+0	9.7e-3	9.5e-3
10^{-6}	20480	497	5.2e-1	6.3e+1	5.3e-2	1.1e-7	1.1e-7
	81920	841	2.1e+0	4.1e+2	2.4e-1	2.3e-7	2.3e-7
	327680	1236	8.2e+0	2.3e+3	1.0e+0	1.2e-6	1.2e-6

Numerical results in 3D

First-kind **volume** integral equation on a **cube** with

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

ϵ	N	$ \hat{c} $	m_f	t_f	$t_{a/s}$	e_a	e_s	n_i
10^{-2}	16^3	39	1.5e-2	1.5e+0	1.5e-2	6.0e-3	2.8e-2	10
	32^3	51	1.7e-1	2.1e+1	1.5e-1	9.0e-3	5.7e-2	14
	64^3	65	1.7e+0	2.8e+2	1.4e+0	1.3e-2	1.3e-1	17
10^{-3}	16^3	92	4.3e-2	2.7e+0	9.6e-3	2.2e-4	1.0e-3	6
	32^3	171	4.1e-1	4.8e+1	5.9e-2	4.0e-4	2.0e-3	8
	64^3	364	4.2e+0	8.8e+2	5.7e-1	7.1e-4	2.4e-3	8
10^{-4}	16^3	182	6.1e-2	3.1e+0	7.2e-3	1.2e-5	1.2e-4	4
	32^3	360	7.7e-1	1.5e+2	8.6e-2	2.8e-5	2.3e-4	5
	64^3	793	9.1e+0	3.5e+3	9.1e-1	5.7e-5	3.6e-4	5

Differential equations

Algorithm: multifrontal

Build quadtree/octree.

for each level $\ell = 0, 1, 2, \dots, L$ from finest to coarsest **do**

 Let C_ℓ be the set of all cells on level ℓ .

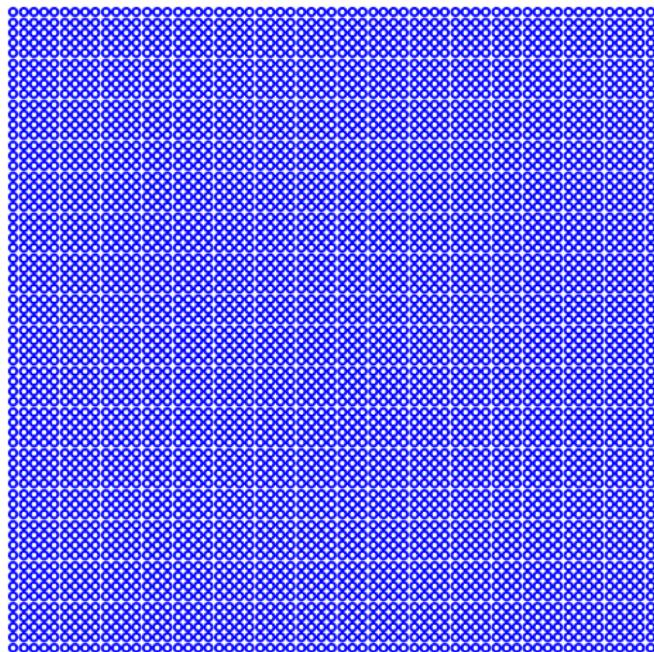
for each cell $c \in C_\ell$ **do**

 Eliminate remaining **interior** DOFs in c .

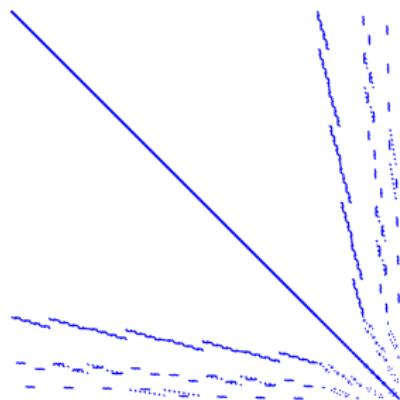
end for

end for

MF in 2D: level 0

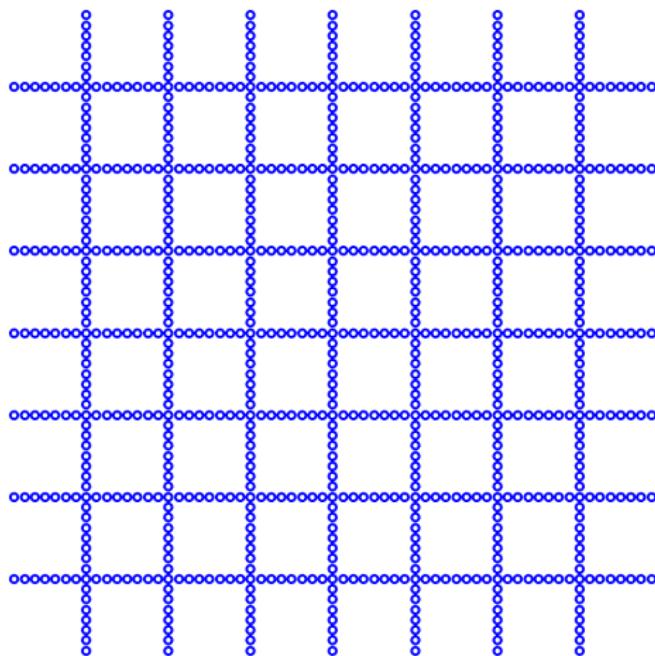


domain

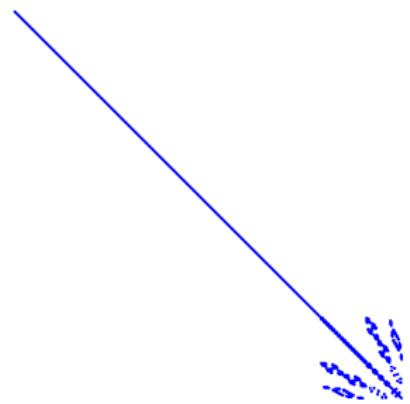


matrix

MF in 2D: level 1

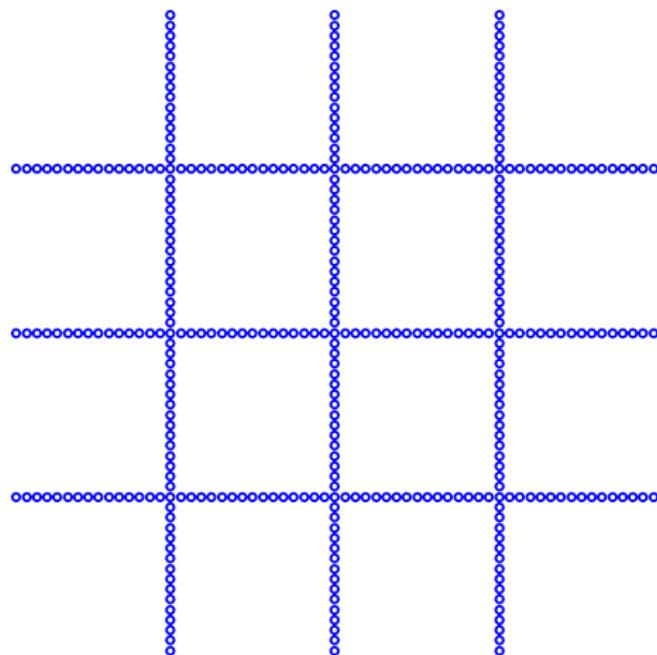


domain

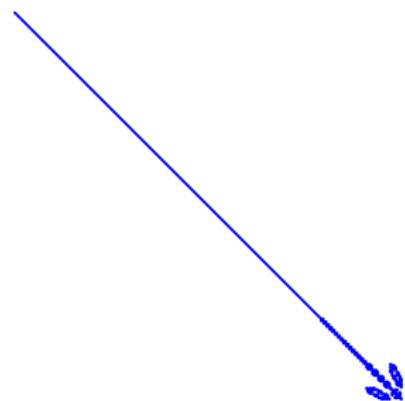


matrix

MF in 2D: level 2

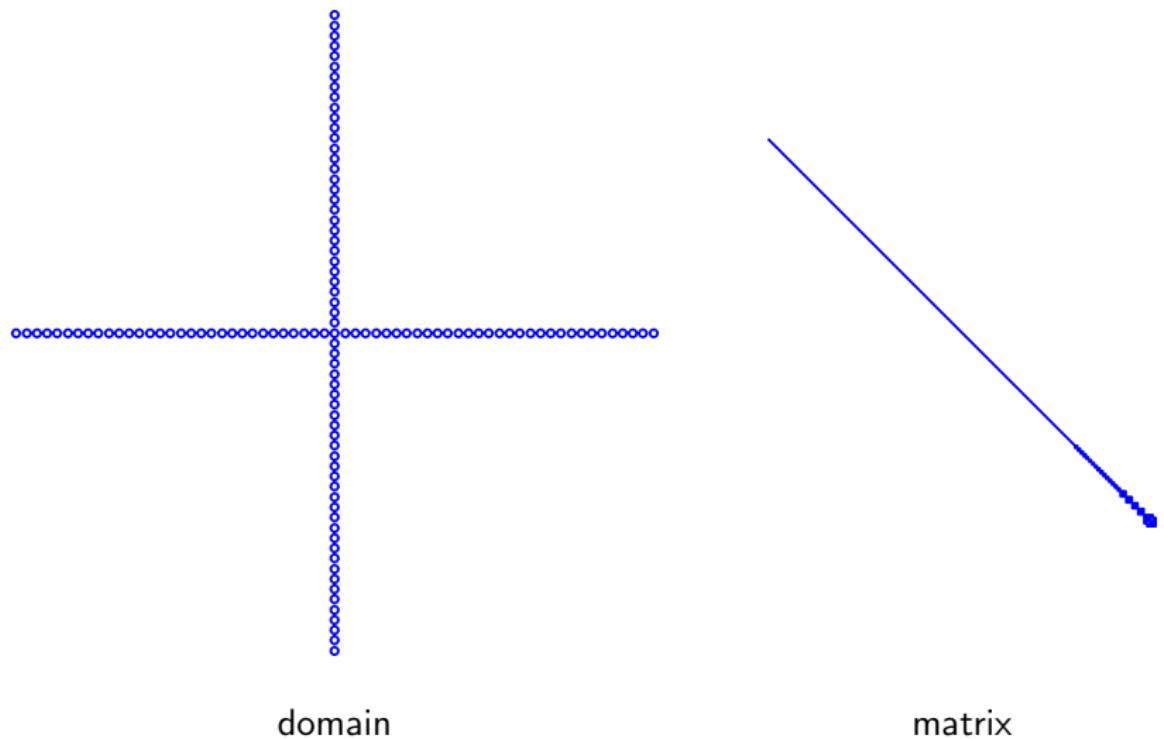


domain

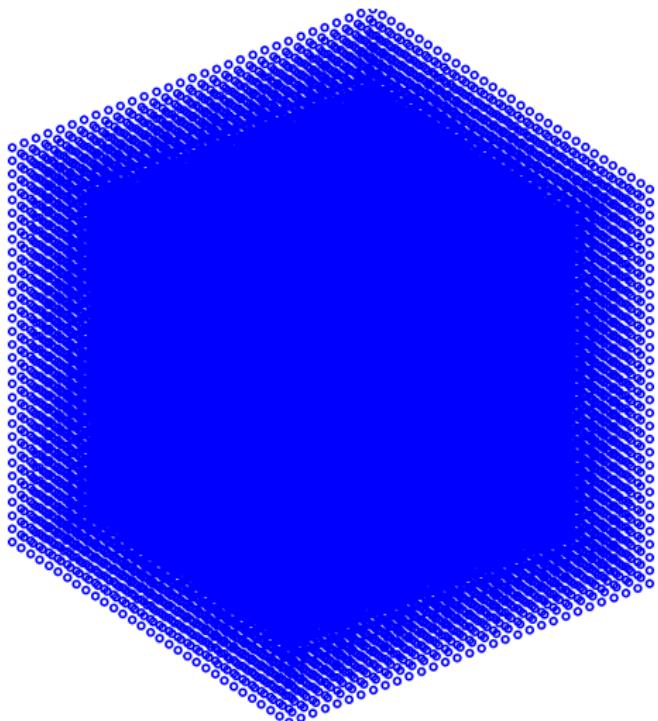


matrix

MF in 2D: level 3

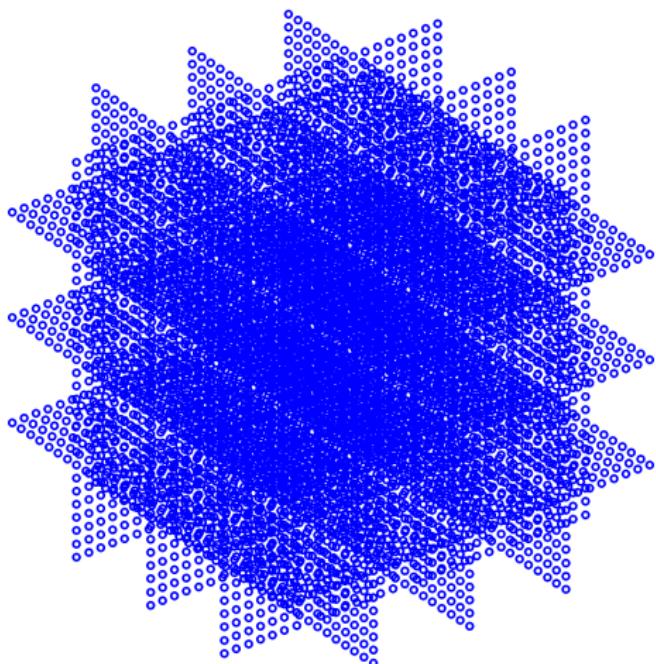


MF in 3D: level 0



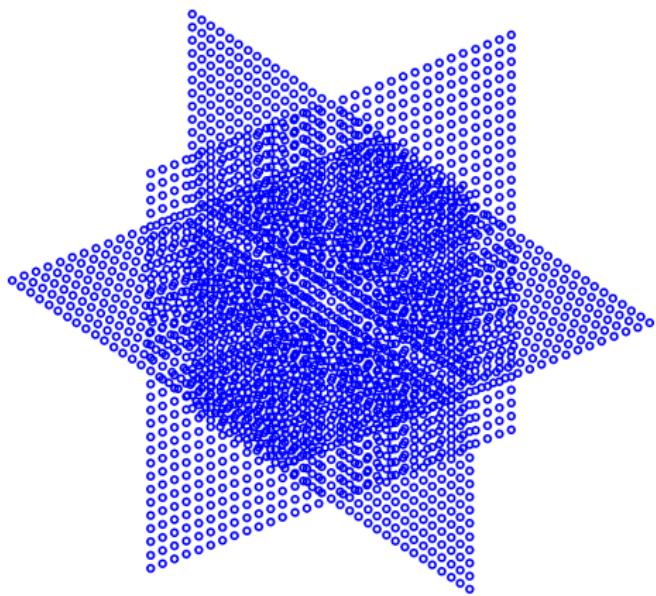
domain

MF in 3D: level 1



domain

MF in 3D: level 2



domain

MF analysis

- ▶ Elimination operator (assume A is Hermitian):

$$W_\ell = \prod_{c \in C_\ell} S_c$$

- ▶ Block diagonalization:

$$D = W_{L-1}^* \cdots W_0^* A W_0 \cdots W_{L-1}$$

- ▶ LU decomposition:

$$\begin{aligned}A &= W_0^{-*} \cdots W_{L-1}^{-*} D W_{L-1}^{-1} \cdots W_0^{-1} \\A^{-1} &= W_0 \cdots W_{L-1} D^{-1} W_L^* \cdots W_0^*\end{aligned}$$

- ▶ Numerically exact: fast direct **solver**

MF analysis

The cost is determined by the separator/front size.

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

Question: How to reduce the front size?

- ▶ Frontal matrices are dense but rank-structured
- ▶ Use integral equation techniques by **skeletonizing** along fronts
- ▶ Dimensional reduction

Algorithm: hierarchical interpolative factorization for PDEs in 2D

Build quadtree.

for each level $\ell = 0, 1, 2, \dots, L$ 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**

Eliminate remaining interior DOFs in c .

end for

 Let $C_{\ell+1/2}$ be the set of all **edges** on level ℓ .

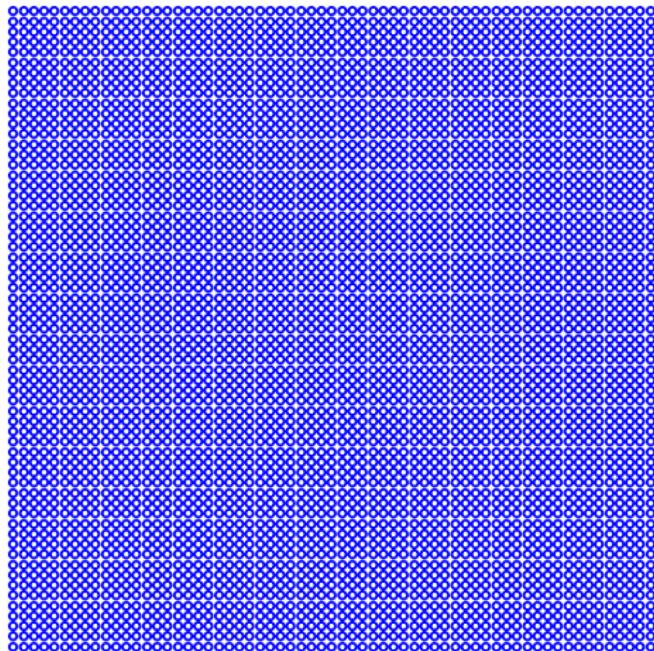
for each cell $c \in C_{\ell+1/2}$ **do**

Skeletonize remaining interior DOFs in c .

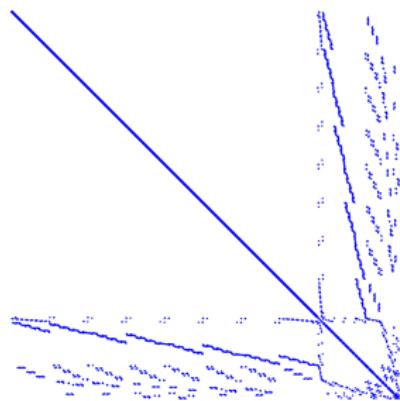
end for

end for

HIF-DE in 2D: level 0

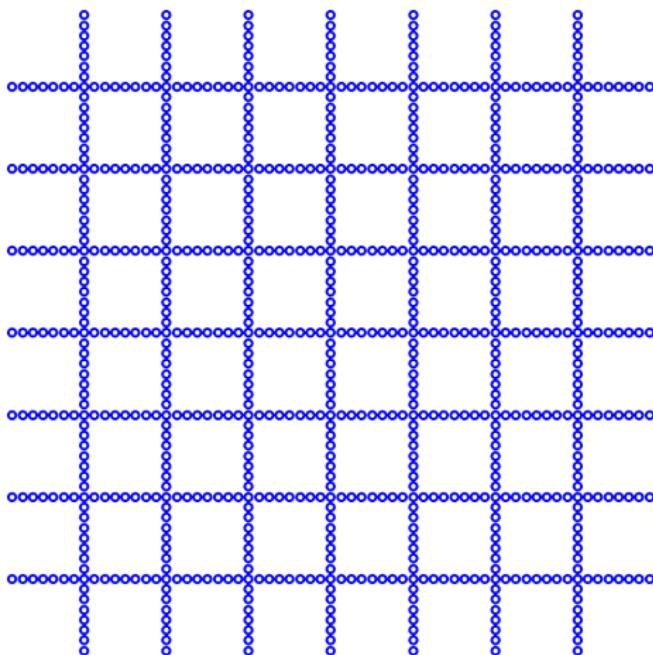


domain

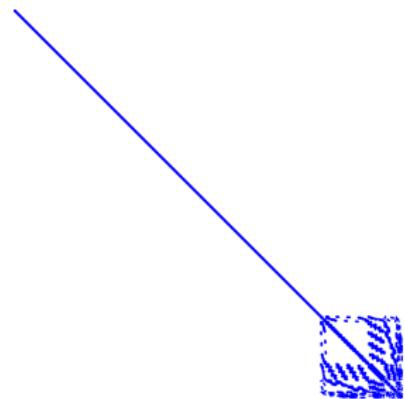


matrix

HIF-DE in 2D: level 1/2

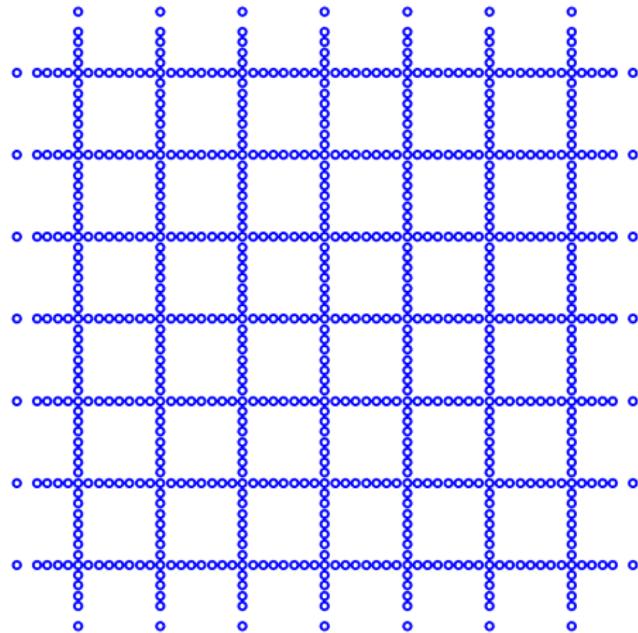


domain

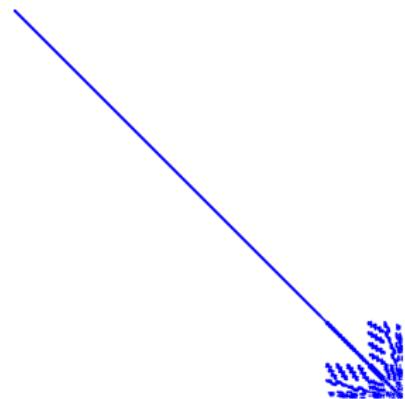


matrix

HIF-DE in 2D: level 1

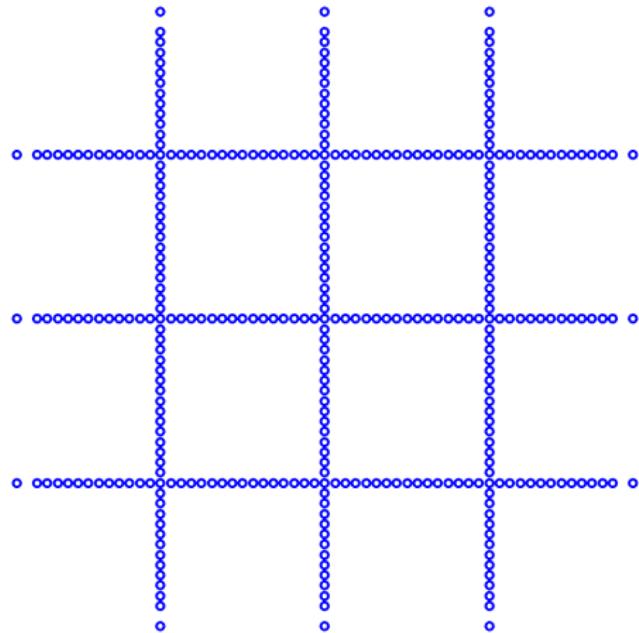


domain

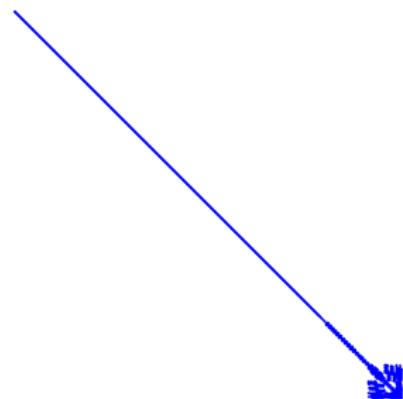


matrix

HIF-DE in 2D: level 3/2

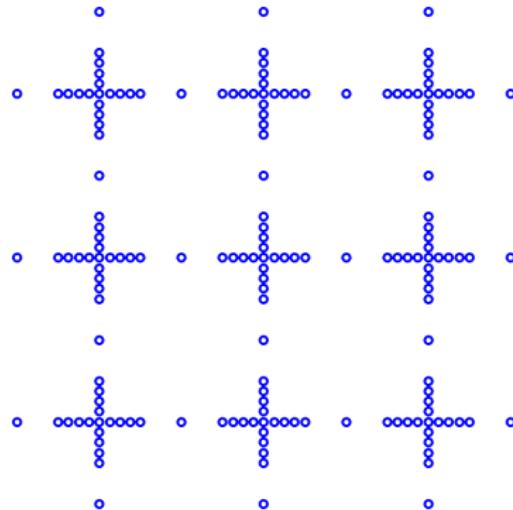


domain

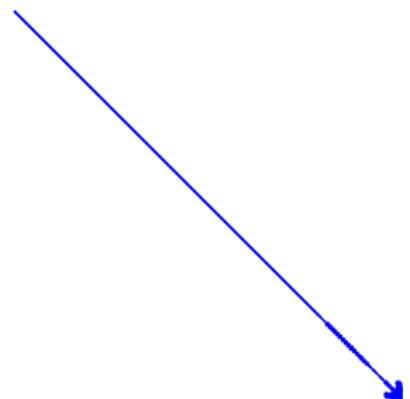


matrix

HIF-DE in 2D: level 2

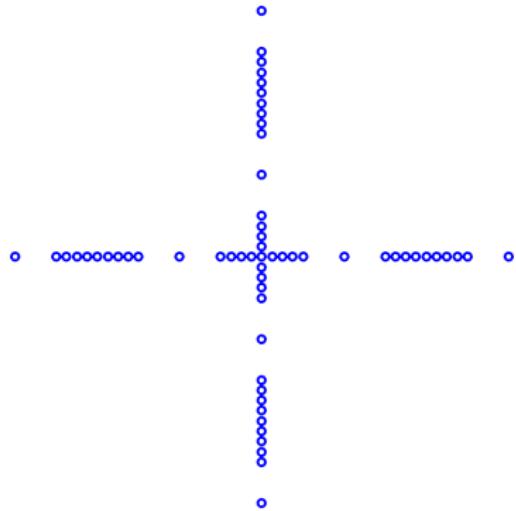


domain

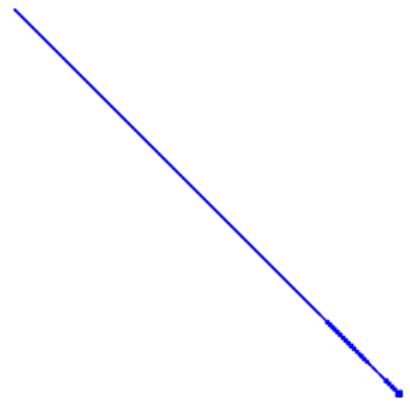


matrix

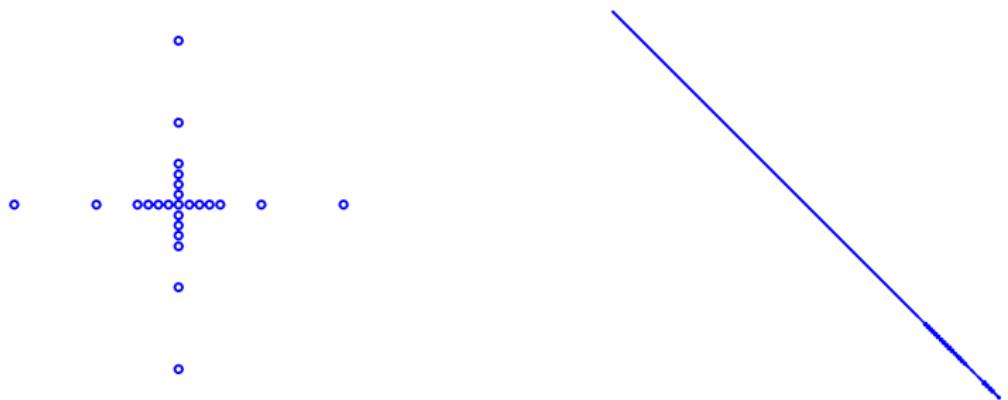
HIF-DE in 2D: level 5/2



domain



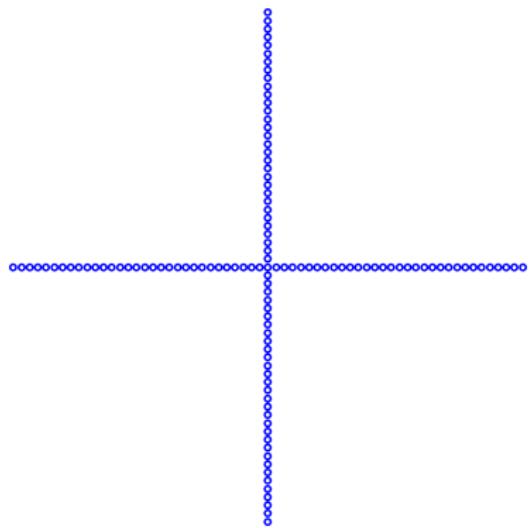
HIF-DE in 2D: level 3



domain

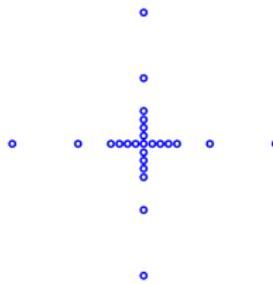
matrix

MF vs. HIF-DE in 2D

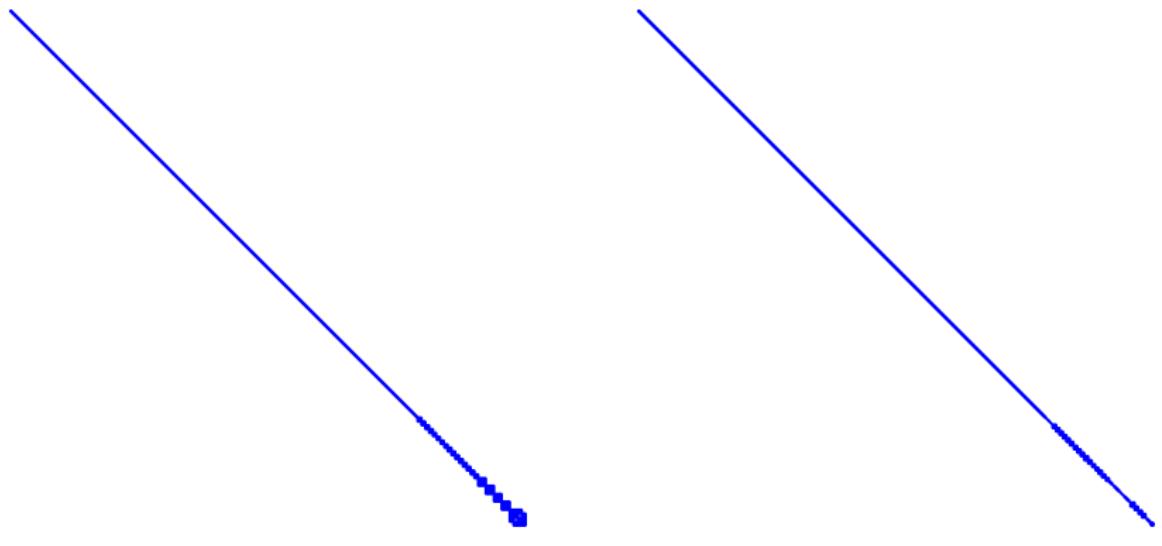


MF

HIF-DE



MF vs. HIF-DE in 2D



MF

HIF-DE

Algorithm: hierarchical interpolative factorization for PDEs in 3D

Build octree.

for each level $\ell = 0, 1, 2, \dots, L$ 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**

Eliminate remaining interior DOFs in c .

end for

 Let $C_{\ell+1/2}$ be the set of all **faces** on level ℓ .

for each cell $c \in C_{\ell+1/2}$ **do**

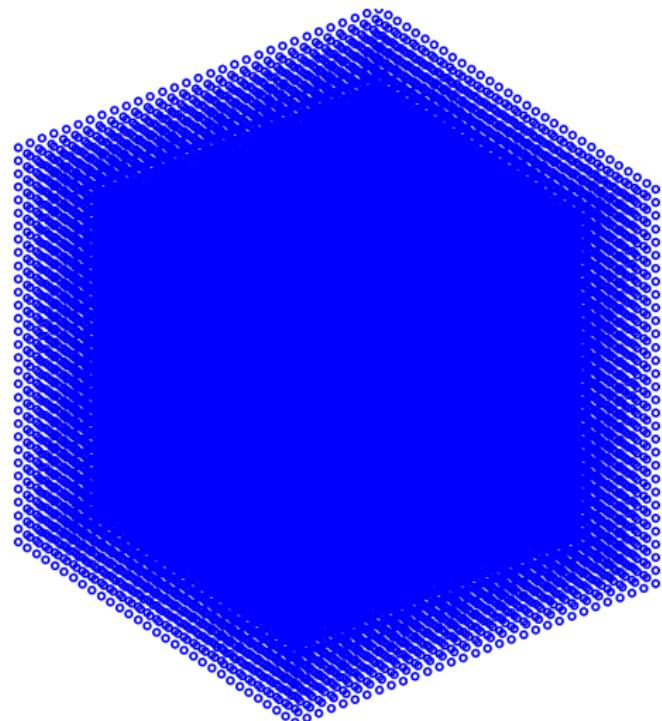
Skeletonize remaining interior DOFs in c .

end for

end for

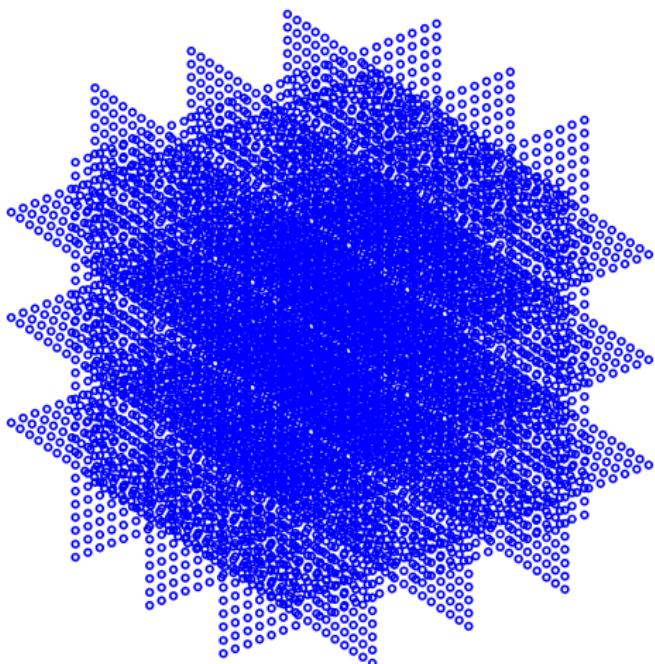
- ▶ In principle, can also skeletonize along edges for true $\mathcal{O}(N)$ complexity
- ▶ This algorithm is sufficient for $\mathcal{O}(N \log N)$ and better exploits sparsity

HIF-DE in 3D: level 0



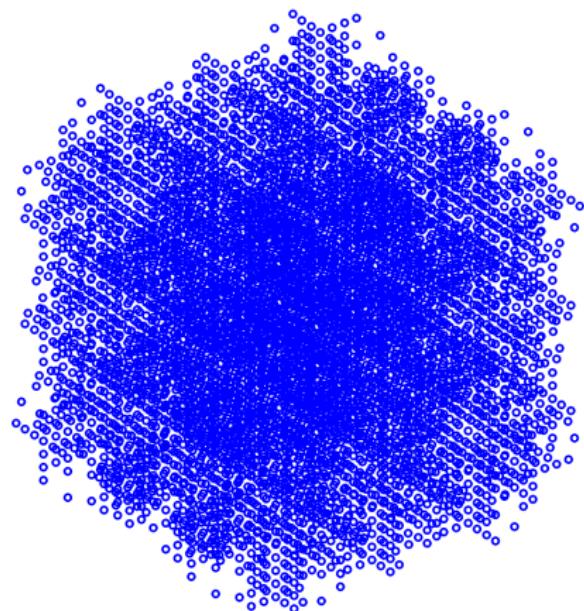
domain

HIF-DE in 3D: level 1/2



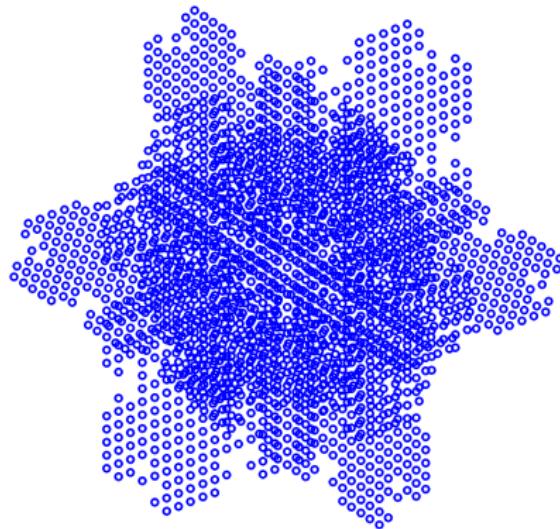
domain

HIF-DE in 3D: level 1



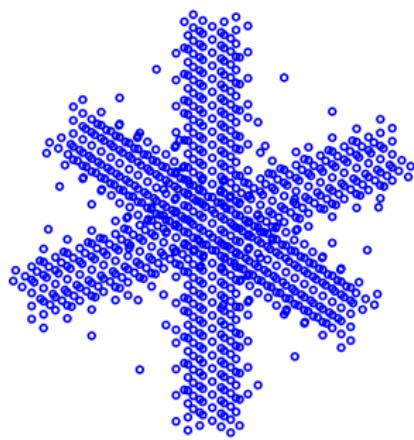
domain

HIF-DE in 3D: level 3/2



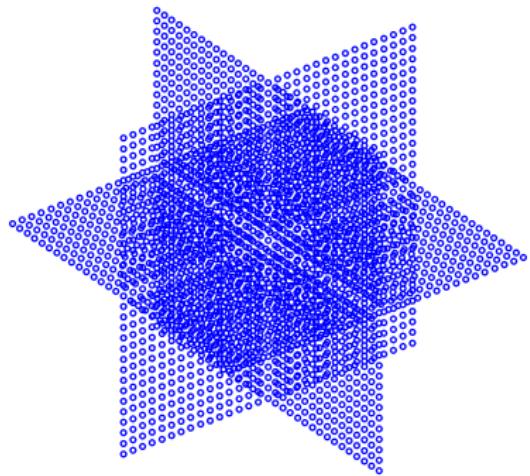
domain

HIF-DE in 3D: level 2

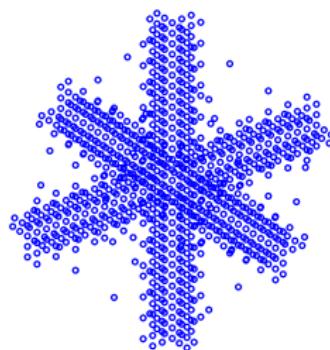


domain

MF vs. HIF-DE in 3D



MF



HIF-DE

HIF-DE analysis

- ▶ Skeletonization operator (assume A is Hermitian):

$$U_\ell = \prod_{c \in C_\ell} Q_c S_c$$

- ▶ Generalized LU decomposition:

$$\begin{aligned} A &\approx W_0^{-*} U_{1/2}^{-*} \cdots W_{L-1}^{-*} U_{L-1/2}^{-*} D U_{L-1/2}^{-1} W_{L-1}^{-1} \cdots U_{1/2}^{-1} W_0^{-1} \\ A^{-1} &\approx W_0 U_{1/2} \cdots W_{L-1} U_{L-1/2} D^{-1} U_{L-1/2}^* W_L^* \cdots U_{1/2}^* W_0^* \end{aligned}$$

- ▶ No longer exact, fast direct solver or preconditioner depending on accuracy

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

Numerical results in 2D

Finite difference discretization on a **square** with

$$a(x) = \prod_{\ell=0}^L \left(\frac{3}{8} \sin(2\pi 2^\ell x_1) \sin(2\pi 2^\ell x_2) + \frac{5}{8} \right), \quad v(x) \equiv 0$$

ϵ	N	$ \hat{c} $	m_f (GB)	t_f (s)	$t_{a/s}$ (s)	e_a	e_s	n_i
10^{-6}	255^2	20	1.4e-1	3.6e+0	1.6e-1	2.3e-08	3.2e-06	3
	511^2	22	5.8e-1	1.7e+1	6.5e-1	2.2e-08	1.1e-05	3
	1023^2	23	2.4e+0	8.1e+1	2.4e+0	2.3e-08	1.8e-05	3
10^{-9}	255^2	31	1.5e-1	3.8e+0	2.1e-1	9.9e-12	1.1e-09	2
	511^2	35	6.0e-1	1.9e+1	6.3e-1	1.5e-11	2.7e-09	2
	1023^2	38	2.4e+0	8.1e+1	2.3e+0	1.6e-11	2.5e-08	2
10^{-12}	255^2	38	1.5e-1	3.5e+0	1.4e-1	1.4e-14	9.9e-13	1
	511^2	44	6.0e-1	1.8e+1	6.2e-1	1.5e-14	6.7e-12	2
	1023^2	50	2.5e+0	9.2e+1	2.6e+0	1.7e-14	7.4e-12	2

Numerical results in 3D

Finite difference discretization on a **cube** with

$$a(x) = \prod_{\ell=0}^L \left(\frac{3}{8} \sin(2\pi 2^\ell x_1) \sin(2\pi 2^\ell x_2) \sin(2\pi 2^\ell x_3) + \frac{5}{8} \right), \quad v(x) \equiv 0$$

ϵ	N	$ \hat{c} $	m_f (GB)	t_f (s)	$t_{a/s}$ (s)	e_a	e_s	n_i
10^{-3}	31^3	83	1.9e-1	6.5e+0	7.4e-2	4.4e-05	5.8e-04	6
	63^3	189	2.1e+0	1.3e+2	8.3e-1	5.1e-05	1.1e-03	7
	127^3	388	2.2e+1	2.0e+3	8.7e+0	6.4e-05	3.2e-03	11
10^{-6}	31^3	152	2.4e-1	8.1e+0	8.5e-2	2.5e-08	1.3e-07	2
	63^3	367	3.1e+0	2.0e+2	1.1e+0	3.1e-08	3.2e-07	3
	127^3	802	3.6e+1	4.1e+3	1.1e+1	4.2e-08	1.3e-06	3
10^{-9}	31^3	197	2.7e-1	8.8e+0	7.9e-2	1.9e-11	6.6e-11	2
	63^3	531	3.7e+0	2.4e+2	1.0e+0	1.8e-11	1.2e-10	2
	127^3	1225	4.6e+1	6.2e+3	1.3e+1	2.7e-11	4.6e-10	2

Conclusions

- ▶ Efficient **factorization** of structured operators in 2D and 3D
 - Fast matrix-vector multiplication
 - Fast direct solver at high accuracy, preconditioner otherwise
 - **Linear complexity** in principle but no proof yet (only empirical results)
- ▶ Dimensional reduction by alternating between cells, faces, and edges
- ▶ Explicit elimination of DOFs, no nested hierarchical operations
- ▶ Can be viewed as adaptive numerical upscaling
- ▶ **Extensions:** $A^{1/2}$, $\log \det A$, $\text{diag } A^{-1}$
- ▶ High accuracy for IEs in 3D still challenging, may require new ideas
- ▶ *Perspective:* structured dense matrices can be sparsified very efficiently
- ▶ Can borrow directly from sparse algorithms, e.g., $RS = MF$
- ▶ What other features of sparse matrices can be exploited?