A fast direct solver for non-oscillatory integral equations

Kenneth L. Ho and Leslie Greengard

Courant Institute, New York University

SIAM AN 2012

Model problem

- Laplace's equation with Dirichlet boundary conditions:

$$\Delta u = 0 \quad \text{in } \Omega \quad , \quad u = f \quad \text{on } \partial\Omega$$

- Fundamental to many areas of mathematical physics
- Solve using a Green's function representation (double-layer potential):

$$u(\mathbf{r}) = \int_{\partial\Omega} \frac{\partial G}{\partial \nu_{\mathbf{s}}}(\mathbf{r}, \mathbf{s})\, \sigma(\mathbf{s})\, dA_{\mathbf{s}} \quad \text{in } \Omega$$

- Integral equation for unknown surface density σ:

$$-\frac{1}{2}\sigma(\mathbf{r}) + \int_{\partial\Omega} \frac{\partial G}{\partial \nu_{\mathbf{s}}}(\mathbf{r}, \mathbf{s})\, \sigma(\mathbf{s})\, dA_{\mathbf{s}} = f(\mathbf{r}) \quad \text{on } \partial\Omega$$

- Discretize: $Ax = b$
- Good: well-conditioned, high-order, dimensional reduction
- Bad: dense matrices, computational cost

Let $A \in \mathbb{C}^{N \times N}$ discretize a non-oscillatory Green's function integral operator.

- Cost of applying A: $\mathcal{O}(N^2)$
- Cost of inverting A: $\mathcal{O}(N^3)$

Fast algorithms are required!

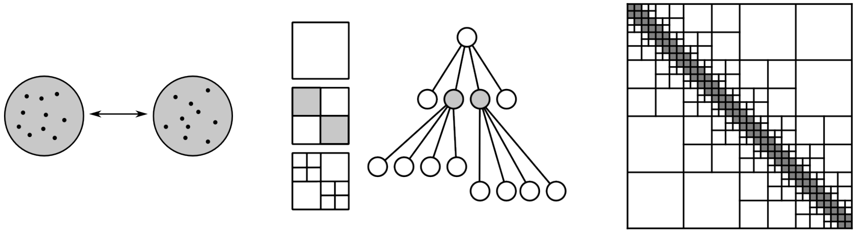
Let $A \in \mathbb{C}^{N \times N}$ discretize a non-oscillatory Green's function integral operator.

- Cost of applying A: $\mathcal{O}(N^2)$
- Cost of inverting A: $\mathcal{O}(N^3)$

Fast algorithms are required!

Fortunately, such matrices are structured.

- Analysis: non-oscillatory Green's functions have smooth far fields
- Algebra: separated off-diagonal matrix blocks are numerically low-rank



- Exploit smoothness with a hierarchical decomposition of space
- $\mathcal{O}(N \log N)$ matrix-vector multiplication (treecode, FMM, panel clustering)
- Combine with GMRES, BiCG, CGR, etc. for fast iterative solvers

Fast iterative solvers have been very successful, but they can still be inefficient:

- When A is ill-conditioned (multiphysics, singular geometries)
- When $Ax = b$ must be solved with many right-hand sides b or many perturbations of a base matrix A (optimization, design, time marching)

Fast iterative solvers have been very successful, but they can still be inefficient:

- When A is ill-conditioned (multiphysics, singular geometries)
- When $Ax = b$ must be solved with many right-hand sides b or many perturbations of a base matrix A (optimization, design, time marching)

One solution: fast direct solvers (construct A^{-1}).

- Robust: insensitive to conditioning, always works
- Fast solves and inverse updates following initial factorization

Fast iterative solvers have been very successful, but they can still be inefficient:

- When A is ill-conditioned (multiphysics, singular geometries)
- When $Ax = b$ must be solved with many right-hand sides b or many perturbations of a base matrix A (optimization, design, time marching)

One solution: fast direct solvers (construct A^{-1}).

- Robust: insensitive to conditioning, always works
- Fast solves and inverse updates following initial factorization

Various approaches in recent years:

- $\mathscr{H}$- and $\mathscr{H}^2$-matrices (Hackbusch, Börm, Grasedyck, Bebendorf et al.)
- HSS matrices (Chandrasekaran, Gu, Xia, Li et al.)
- Skeletonization (Martinsson, Rokhlin, Gillman, Greengard, Bremer et al.)
  - BIEs in 2D
  - One-level BIEs in 3D

Here, we present a multilevel skeletonization-based fast direct solver in general dimension. For BIEs:

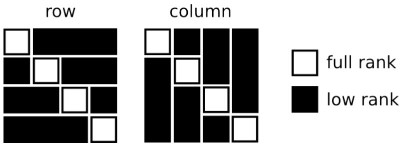
	2D	3D
precomp	$\mathcal{O}(N)$	$\mathcal{O}(N^{3/2})$
solve	$\mathcal{O}(N)$	$\mathcal{O}(N \log N)$

Main ideas/take-home messages :

- Kernel-independent: Laplace, Stokes, Yukawa, low-frequency Helmholtz
- Robust to geometry (e.g., boundary vs. volume, dimensionality)
- User-specified precision: trade accuracy for speed
- Naturally exposes the underlying sparsity of integral equation matrices
- Transparently takes advantage of sparse direct solver development
- Very fast solve times, beating the FMM by factors of 100–1000
- Simple framework: easy to analyze, implement, and optimize
- Somewhat similar in flavor to nested dissection

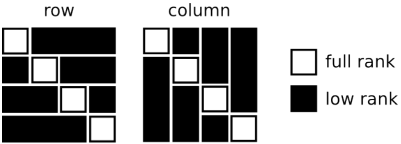
A block matrix A is block separable if

$$\underbrace{\begin{bmatrix} \times & \times \\ \times & \times \end{bmatrix}}_{A_{ij}} = \underbrace{\begin{bmatrix} \times \\ \times \end{bmatrix}}_{L_i} \underbrace{\begin{bmatrix} \times \end{bmatrix}}_{S_{ij}} \underbrace{\begin{bmatrix} \times & \times \end{bmatrix}}_{R_j} \quad , \qquad i \neq j.$$

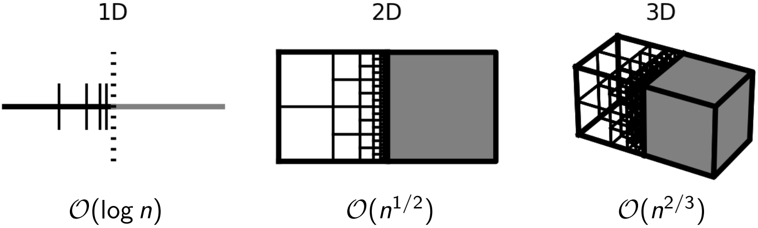


A block matrix A is block separable if

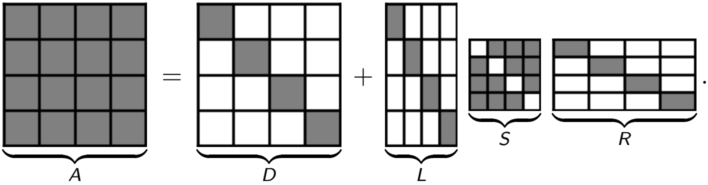
$$\underbrace{\begin{bmatrix}\times & \times\\ \times & \times\end{bmatrix}}_{A_{ij}} = \underbrace{\begin{bmatrix}\times\\ \times\end{bmatrix}}_{L_i}\underbrace{\begin{bmatrix}\times\end{bmatrix}}_{S_{ij}}\underbrace{\begin{bmatrix}\times & \times\end{bmatrix}}_{R_j}, \qquad i \neq j.$$



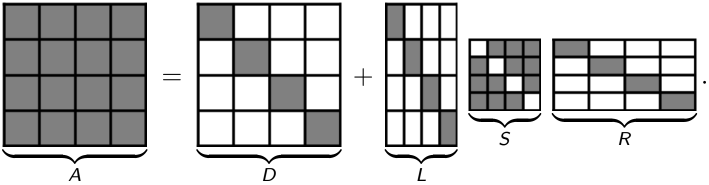
Integral equation matrices are block separable.



If A is block separable, then



If A is block separable, then



The inverse can be written in essentially the same form:

$$A^{-1} = \mathcal{D} + \mathcal{L}\mathcal{S}^{-1}\mathcal{R},$$

where

$$\mathcal{D} = D^{-1} - D^{-1}L\Lambda RD^{-1}, \quad \mathcal{L} = D^{-1}L\Lambda, \quad \mathcal{R} = \Lambda RD^{-1}, \quad \mathcal{S} = \Lambda + S,$$

with $\Lambda = \left(RD^{-1}L\right)^{-1}$.

- $\mathcal{D}$, $\mathcal{L}$, and $\mathcal{R}$ are block diagonal
- Reduces to inverting $\mathcal{S} \in \mathbb{C}^{K\times K}$, where typically $K \ll N$

We can also adopt a sparse matrix perspective. For

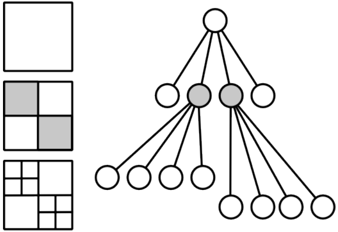
$$Ax = (D + LSR)\,x = b,$$

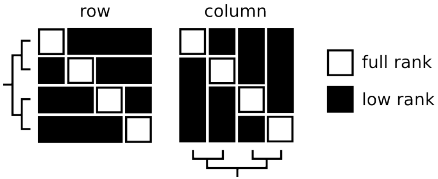
let $z \equiv Rx$ and $y \equiv Sz$. Then this is equivalent to the structured sparse system

$$\begin{bmatrix} D & L & \\ R & & -I \\ & -I & S \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}.$$

Factor using UMFPACK, SuperLU, MUMPS, Pardiso, etc.

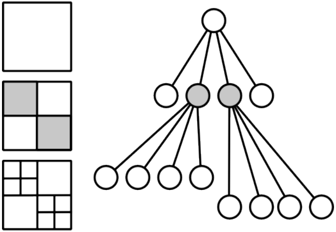
Integral equation matrices are, in fact, hierarchically block separable, i.e., they are block separable at every level of an octree-type ordering.

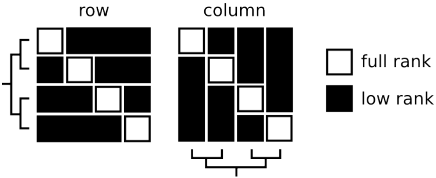




In this setting, much more powerful algorithms can be developed.

Integral equation matrices are, in fact, hierarchically block separable, i.e., they are block separable at every level of an octree-type ordering.





In this setting, much more powerful algorithms can be developed.

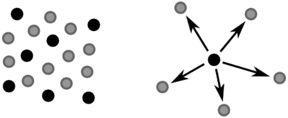
How to compress to block separable form?

An interpolative decomposition of a rank-k matrix is a representation

$$\underbrace{A}_{m\times n} = \underbrace{B}_{m\times k}\underbrace{P}_{k\times n},$$

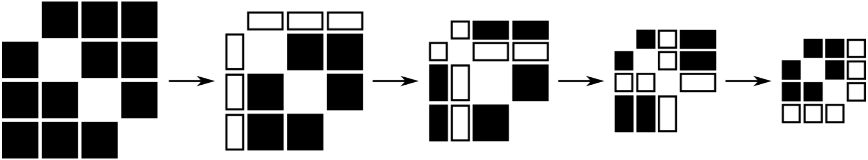
where B is a column-submatrix of A (with $\|P\|$ small).

- The ID compresses the column space; to compress the row space, apply the ID to A^{T}. We call the retained rows and columns skeletons.
- Adaptive algorithms can compute the ID to any specified precision $\epsilon > 0$.
- Related factorizations: RRQR, pseudoskeleton (CUR), ACA



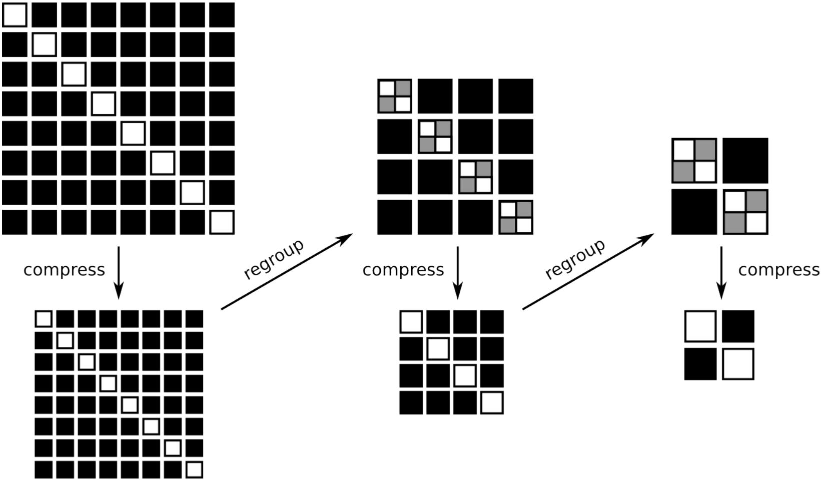
One-level matrix compression

- Compress the row space of each off-diagonal block row.
  Let the L_i be the corresponding row interpolation matrices.
- Compress the column space of each off-diagonal block column.
  Let the R_j be the corresponding column interpolation matrices.
- Approximate the off-diagonal blocks by $A_{ij} \approx L_i S_{ij} R_j$ for $i \neq j$.
- S is a skeleton submatrix of A

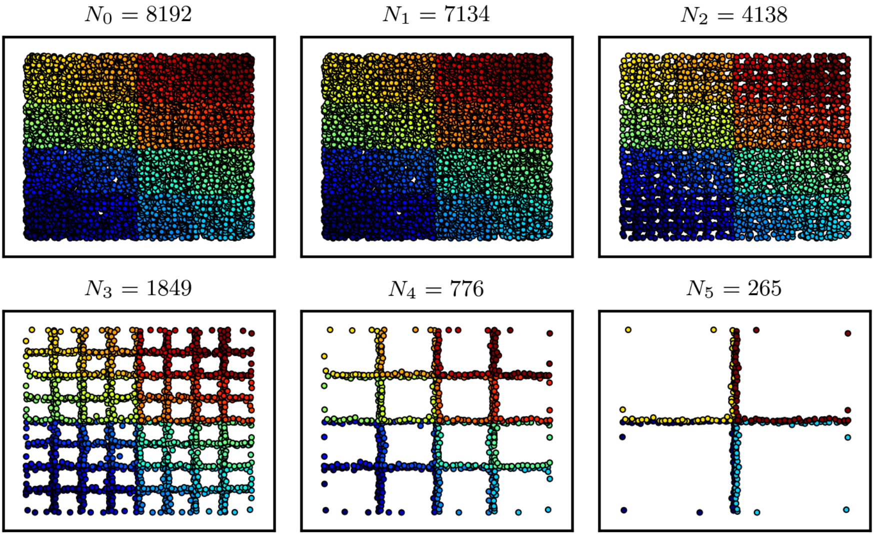


Skeletonization

Multilevel matrix compression



Recursive skeletonization



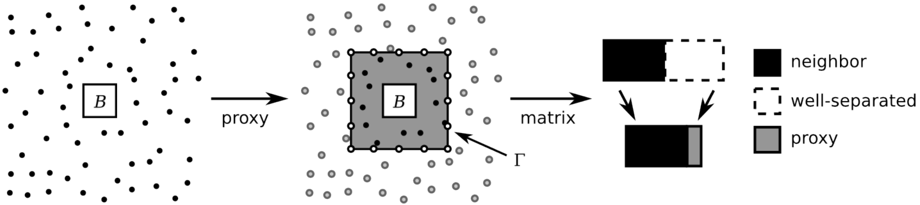
$$G(\mathbf{r}, \mathbf{s}) = -\frac{1}{2\pi} \log |\mathbf{r} - \mathbf{s}| \ , \quad \epsilon = 10^{-3}$$

Accelerated compression for PDEs

- General compression algorithm is global and so at least $\mathcal{O}(N^2)$
- For potential fields, use Green's theorem to accelerate:

$$u(\mathbf{r}) = \int_{\Gamma} \left[u(\mathbf{s}) \frac{\partial G}{\partial \nu_{\mathbf{s}}}(\mathbf{r}, \mathbf{s}) - G(\mathbf{r}, \mathbf{s}) \frac{\partial u}{\partial \nu}(\mathbf{s}) \right] dA_{\mathbf{s}}$$

- Represent well-separated points with a local proxy surface



Compressed matrix representation

Telescoping formula:

$$A \approx D^{(1)} + L^{(1)} \left[D^{(2)} + L^{(2)} \left(\cdots D^{(\lambda)} + L^{(\lambda)} S R^{(\lambda)} \cdots \right) R^{(2)} \right] R^{(1)}$$

- Efficient storage (data-sparse)

N	uncomp	comp
8192	537 MB	9.7 MB
131072	137 GB	184 MB

- Fast matrix-vector multiplication (generalized FMM)
- Fast matrix factorization and inverse application

Compressed matrix inversion

Recursively expand in sparse form:

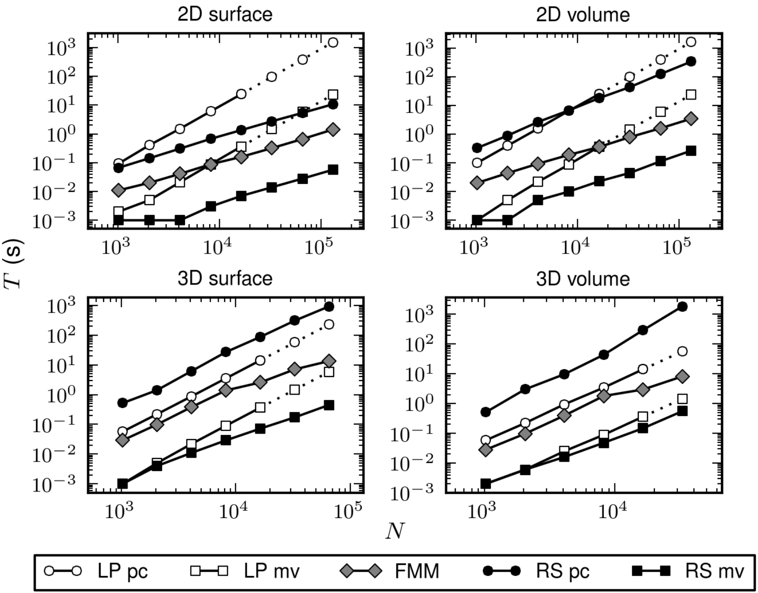
$$\begin{bmatrix} D^{(1)} & L^{(1)} & & & & & \\ R^{(1)} & & -I & & & & \\ & -I & D^{(2)} & L^{(2)} & & & \\ & & R^{(2)} & \ddots & \ddots & & \\ & & & \ddots & D^{(\lambda)} & L^{(\lambda)} & \\ & & & & R^{(\lambda)} & & -I \\ & & & & & -I & S \end{bmatrix} \begin{bmatrix} x \\ y^{(1)} \\ z^{(1)} \\ \vdots \\ \vdots \\ y^{(\lambda)} \\ z^{(\lambda)} \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

This can be treated efficiently using any standard sparse direct solver.

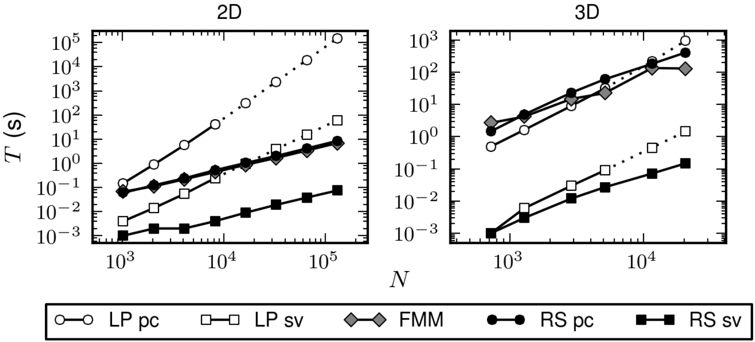
Multilevel inversion formula (for analysis):

$$A^{-1} \approx \mathcal{D}^{(1)} + \mathcal{L}^{(1)} \left[\mathcal{D}^{(2)} + \mathcal{L}^{(2)} \left(\cdots D^{(\lambda)} + \mathcal{L}^{(\lambda)} \mathcal{S}^{-1} \mathcal{R}^{(\lambda)} \cdots \right) \mathcal{R}^{(2)} \right] \mathcal{R}^{(1)}$$

Laplace FMM


2D surface
2D volume
3D surface
3D volume
T (s)
N
LP pc
LP mv
FMM
RS pc
RS mv

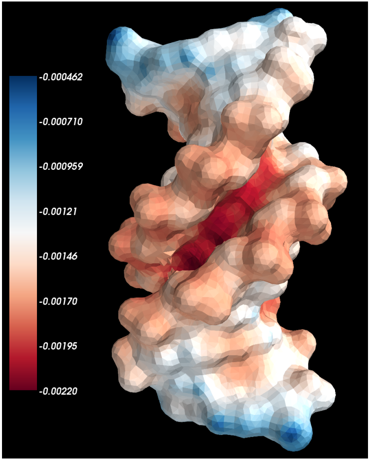
Laplace BIE solver



- Less memory-efficient than FMM/GMRES
- Each solve is extremely fast (in elements/sec)

ϵ	10^{-3}	10^{-6}	10^{-9}
2D	3.3×10^6	2.0×10^6	1.7×10^6
3D	6.0×10^5	1.4×10^5	6.2×10^4

Poisson electrostatics

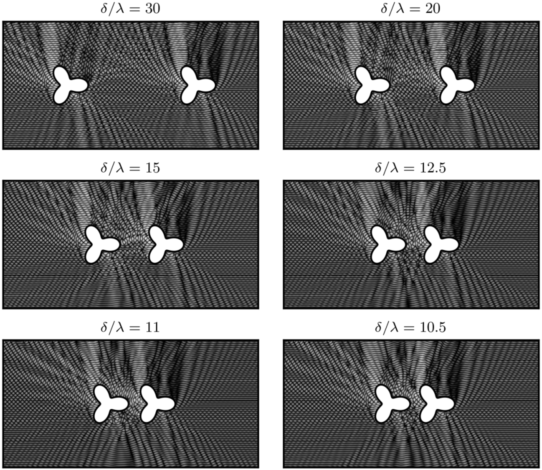


$$
\begin{aligned}
-\Delta\varphi &= 0 && \text{in } \Omega_0 \\
-\Delta\varphi &= \frac{1}{\varepsilon_1}\sum_i q_i \delta\left(\mathbf{r} - \mathbf{r}_i\right) && \text{in } \Omega_1 \\
[\varphi] = \left[\varepsilon \frac{\partial \varphi}{\partial \nu}\right] &= 0 && \text{on } \Sigma
\end{aligned}
$$

N	7612	19752
FMM/GMRES	12.6 s	26.9 s
RS precomp	151 s	592 s
RS solve	0.03 s	0.08 s

Break-even point: 10–25 solves

Multiple scattering



- Each object: 10λ

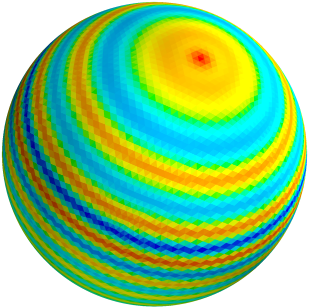
$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

- FMM/GMRES with block preconditioner via RS

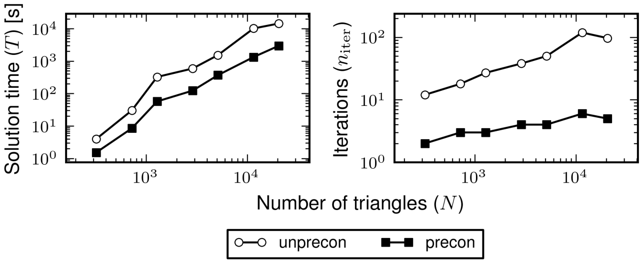
$$\begin{bmatrix} A_{11}^{-1} & \\ & A_{22}^{-1} \end{bmatrix}$$

- Unprecon: 700 iterations
- Precon: 10 iterations
- $50\times$ speedup

Helmholtz preconditioning



- $N = 20480$, 10λ
- Precondition with low-precision inverse ($\epsilon = 10^{-3}$)
- Iterate for full accuracy ($\epsilon = 10^{-12}$)
- Unprecon: 190 iterations
- Precon: 6 iterations
- $10\times$ speedup



Summary

Complexities in d dimensions (BIEs in $d+1$ dimensions):

$$\text{precomp} \sim \begin{cases} N & \text{if } d = 1, \\ N^{3(1-1/d)} & \text{if } d > 1, \end{cases} \qquad \text{solve} \sim \begin{cases} N & \text{if } d = 1, \\ N \log N & \text{if } d = 2, \\ N^{2(1-1/d)} & \text{if } d > 2 \end{cases}$$

- Mild assumptions: low-rank off-diagonal blocks, Green's theorem
- Based only on numerical linear algebra
- Kernel-independent: Laplace, Stokes, Yukawa, low-frequency Helmholtz
- Very fast solves following precomputation (~ 0.1 s)
- Highly effective for preconditioning
- Reveals connection with sparse matrices (Chandrasekaran, Gu et al.)
- Naturally parallelizable via block-sweep structure
- Extensions: low-rank updates, least squares, other matrix decompositions
- Similar ideas also apply for PDE formulations (Xia, Gillman et al.)