A fast direct solver for non-oscillatory integral equations

Kenneth L. Ho

Courant Institute, New York University

Argonne Lab, Jan 2012

Outline

A model problem

Consider Laplace's equation with Dirichlet boundary conditions:

$$\Delta u = 0 \quad \text{in } \Omega \subset \mathbb{R}^3 \quad , \quad u = f \quad \text{on } \partial\Omega.$$

How to find u?

A model problem

Consider Laplace's equation with Dirichlet boundary conditions:

$$\Delta u = 0 \quad \text{in } \Omega \subset \mathbb{R}^3 \quad , \quad u = f \quad \text{on } \partial\Omega.$$

How to find u? Use a double-layer potential representation

$$u(x) = \int_{\partial\Omega} \frac{\partial G}{\partial \nu(y)}(x,y)\,\sigma(y)\,ds(y) \quad \text{in } \Omega,$$

where $G(x,y) = 1/(4\pi|x-y|)$ is the Laplace Green's function, ν is the unit outer surface normal, and σ is an unknown surface density.

A model problem

Consider Laplace's equation with Dirichlet boundary conditions:

$$\Delta u = 0 \quad \text{in } \Omega \subset \mathbb{R}^3 \quad , \quad u = f \quad \text{on } \partial\Omega.$$

How to find u? Use a double-layer potential representation

$$u(x) = \int_{\partial\Omega} \frac{\partial G}{\partial \nu(y)}(x,y)\,\sigma(y)\,ds(y) \quad \text{in } \Omega,$$

where $G(x,y) = 1/(4\pi|x-y|)$ is the Laplace Green's function, ν is the unit outer surface normal, and σ is an unknown surface density. Then appeal to classical potential theory to derive the integral equation

$$-\frac{1}{2}\sigma(x) + \int_{\partial\Omega} \frac{\partial G}{\partial \nu(y)}(x,y)\,\sigma(y)\,ds(y) = f(x) \quad \text{on } \partial\Omega.$$

Why integral equations?

- Well-conditioned (second-kind Fredholm equation)
- Accurate derivatives (differentiate under the integral)
- High-order methods (quadrature)
- Adaptive (often with dimensional reduction)

Why integral equations?

- Well-conditioned (second-kind Fredholm equation)
- Accurate derivatives (differentiate under the integral)
- High-order methods (quadrature)
- Adaptive (often with dimensional reduction)

Mathematically, the "right" thing to do: try to write down as much of the solution as possible before turning to numerics.

An exaggerated example

Consider Poisson's equation with free-space boundary conditions:

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^3 \quad , \quad u = \mathcal{O}\left(\frac{1}{|x|}\right) \quad \text{as } |x| \to \infty.$$

An exaggerated example

Consider Poisson's equation with free-space boundary conditions:

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^3 \quad , \quad u = \mathcal{O}\left(\frac{1}{|x|}\right) \quad \text{as } |x| \to \infty.$$

Invert the differential operator:

$$u(x) = \left[(-\Delta)^{-1} f\right](x) = \int_\Omega G(x,y) f(y)\, dv(y).$$

In this case, there is no integral equation; the solution is just an integral.

An exaggerated example

Consider Poisson's equation with free-space boundary conditions:

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^3 \quad , \quad u = \mathcal{O}\left(\frac{1}{|x|}\right) \quad \text{as } |x| \to \infty.$$

Invert the differential operator:

$$u(x) = \left[(-\Delta)^{-1} f\right](x) = \int_\Omega G(x,y) f(y)\, dv(y).$$

In this case, there is no integral equation; the solution is just an integral.

Procedure for general boundary conditions

- Integrate volume term
- Solve for boundary correction

Numerical considerations

Let $A \in \mathbb{C}^{N \times N}$ be a matrix discretization of some Green's function integral operator. (How to discretize?) Observe that A is dense.

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

Numerical considerations

Let $A \in \mathbb{C}^{N \times N}$ be a matrix discretization of some Green's function integral operator. (How to discretize?) Observe that A is dense.

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

This is contrast to competing methods based on finite differences or finite elements, which yield sparse matrices. Historically, this relative expense was a primary cause for the dearth of integral equations in numerical computing (except where there was no choice).

Fast iterative solvers

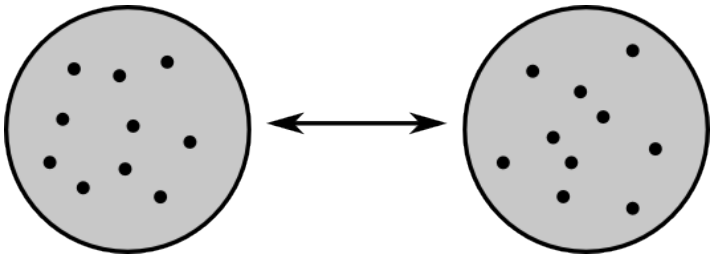
In the 1980s, fast algorithms to apply A in only $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ time were developed:

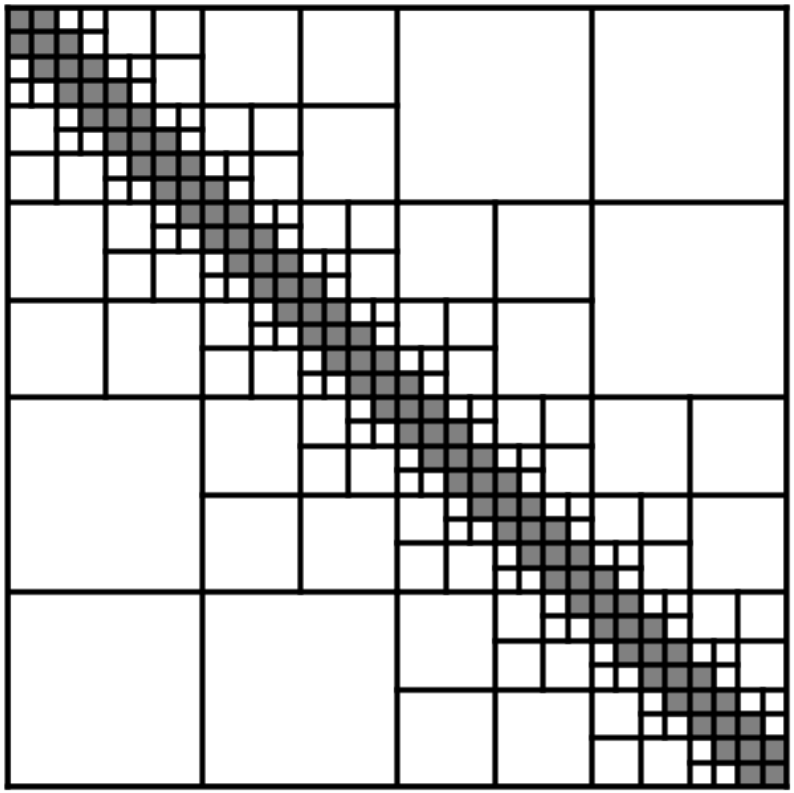
- Treecode (Barnes and Hut, 1986)
- Fast multipole method (Greengard and Rokhlin, 1987)
- Panel clustering (Hackbusch and Nowak, 1989)

Combined with Krylov methods (e.g., GMRES), such techniques enabled fast iterative solution with only $\mathcal{O}(N \log N)$ work in many situations.

Fast multipole (and related) methods

- Non-oscillatory Green's functions have smooth far fields
- Interactions between well-separated clusters are low-rank
- Exploit smoothness with a hierarchical decomposition of space





Still more work to do

Fast iterative solvers have been very successful, but they remain inefficient in certain important regimes:

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

Still more work to do

Fast iterative solvers have been very successful, but they remain inefficient in certain important regimes:

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

One solution: direct solvers.

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

Still more work to do

Fast iterative solvers have been very successful, but they remain inefficient in certain important regimes:

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

One solution: direct solvers.

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

Can we accelerate direct solvers to the same extent?

Fast direct solvers

Much active research in recent years:

- $\mathcal{H}$-matrices (Hackbusch et al., 1999, 2000, 2002)
- Lippmann-Schwinger in 2D (Chen, 2002)
- FMM structure (Pals, 2004)
- BIEs in 2D (Martinsson and Rokhlin, 2005)
- HSS matrices (Chandrasekaran et al., 2006)
- one-level BIE solver in 3D (Greengard et al., 2009)

Current state of the art

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

Block-separable matrices

Definition

A block matrix A is block-separable if

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

Block-separable matrices

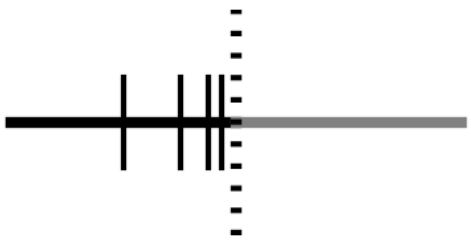
Definition

A block matrix A is block-separable if

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

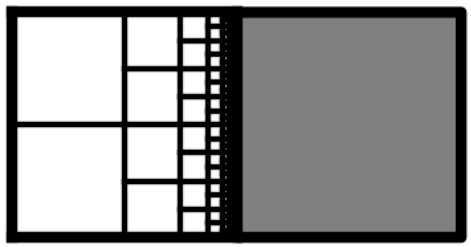
Integral equation matrices are block-separable.

1D



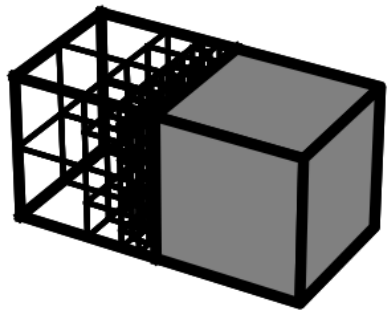
$\mathcal{O}(\log n)$

2D



$\mathcal{O}(n^{1/2})$

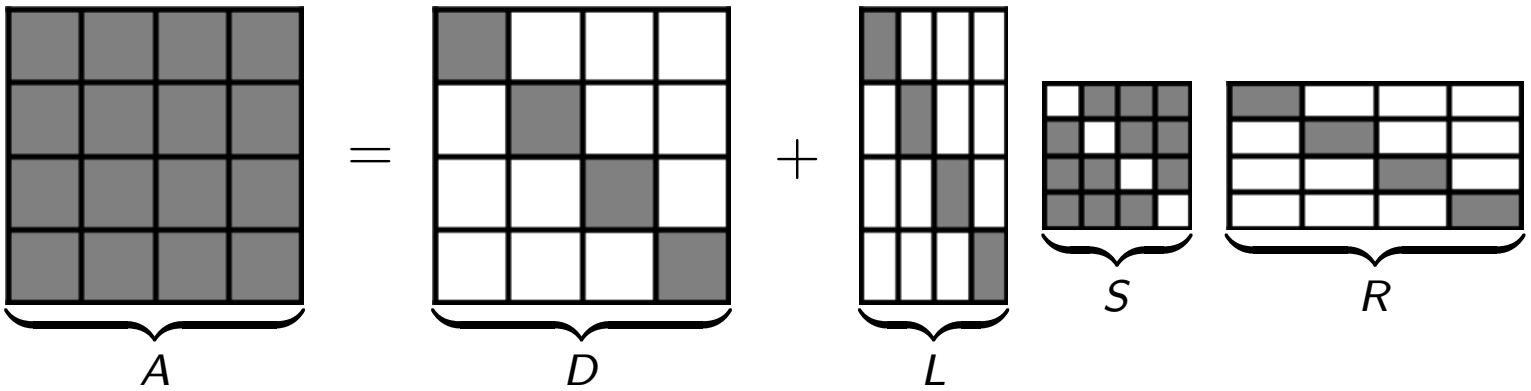
3D



$\mathcal{O}(n^{2/3})$

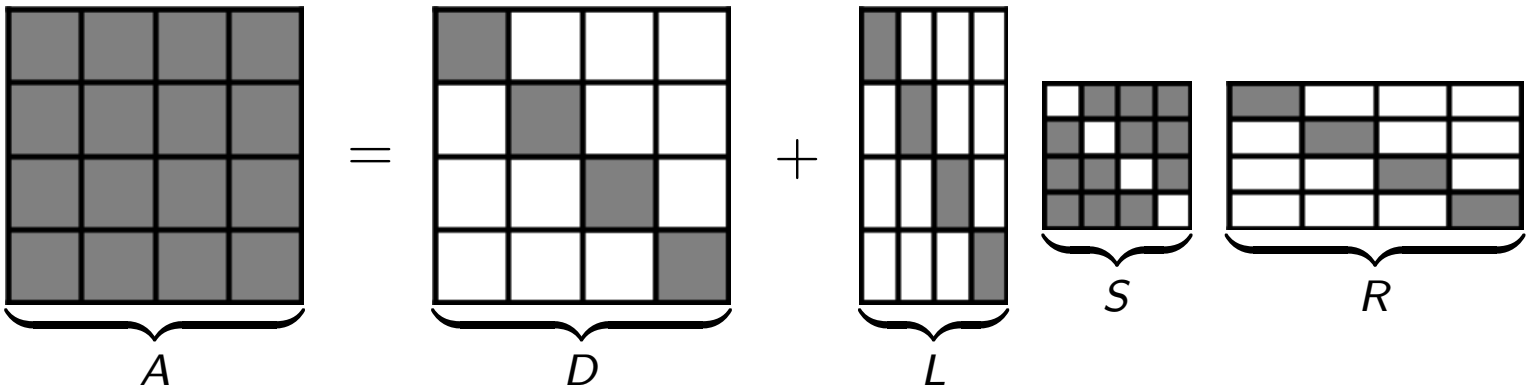
A fast direct solver for block-separable matrices

If A is block-separable, then



A fast direct solver for block-separable matrices

If A is block-separable, then



and $A^{-1} = \mathcal{D} + \mathcal{L}\mathcal{S}^{-1}\mathcal{R}$, 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 $\Lambda = \left(RD^{-1}L\right)^{-1}$. If A has $p \times p$ blocks and each $S_{ij} \in \mathbb{C}^{k\times k}$, then A^{-1} can be computed in $\mathcal{O}(p(N/p)^3 + (pk)^3)$ operations.

A sparse matrix perspective

Consider the system

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

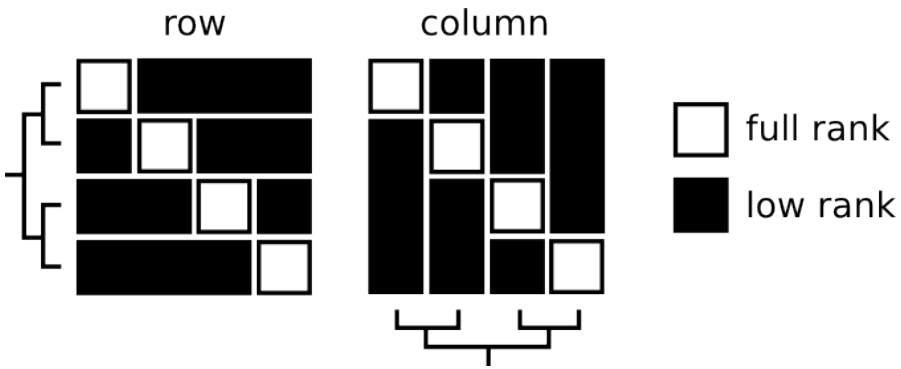
With $z = Rx$ and $y = Sz$, this is equivalent to the structured sparse system

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

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

Hierarchically block-separable matrices

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.

Interpolative decomposition

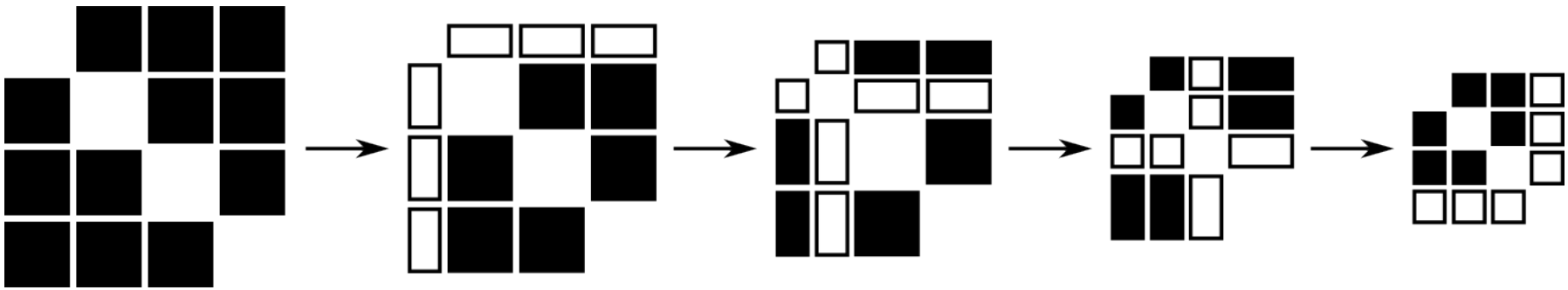
Definition

An interpolative decomposition of a rank-k matrix $A \in \mathbb{C}^{m \times n}$ is a representation $A = BP$, where $B \in \mathbb{C}^{m \times k}$ is a column-submatrix of A and $P \in \mathbb{C}^{k \times n}$ contains the $k \times k$ identity, such that $\|P\|$ is 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 exist to compute the ID to any precision $\epsilon > 0$ in $\mathcal{O}(kmn)$ time (Cheng et al., 2005; Liberty et al., 2007).

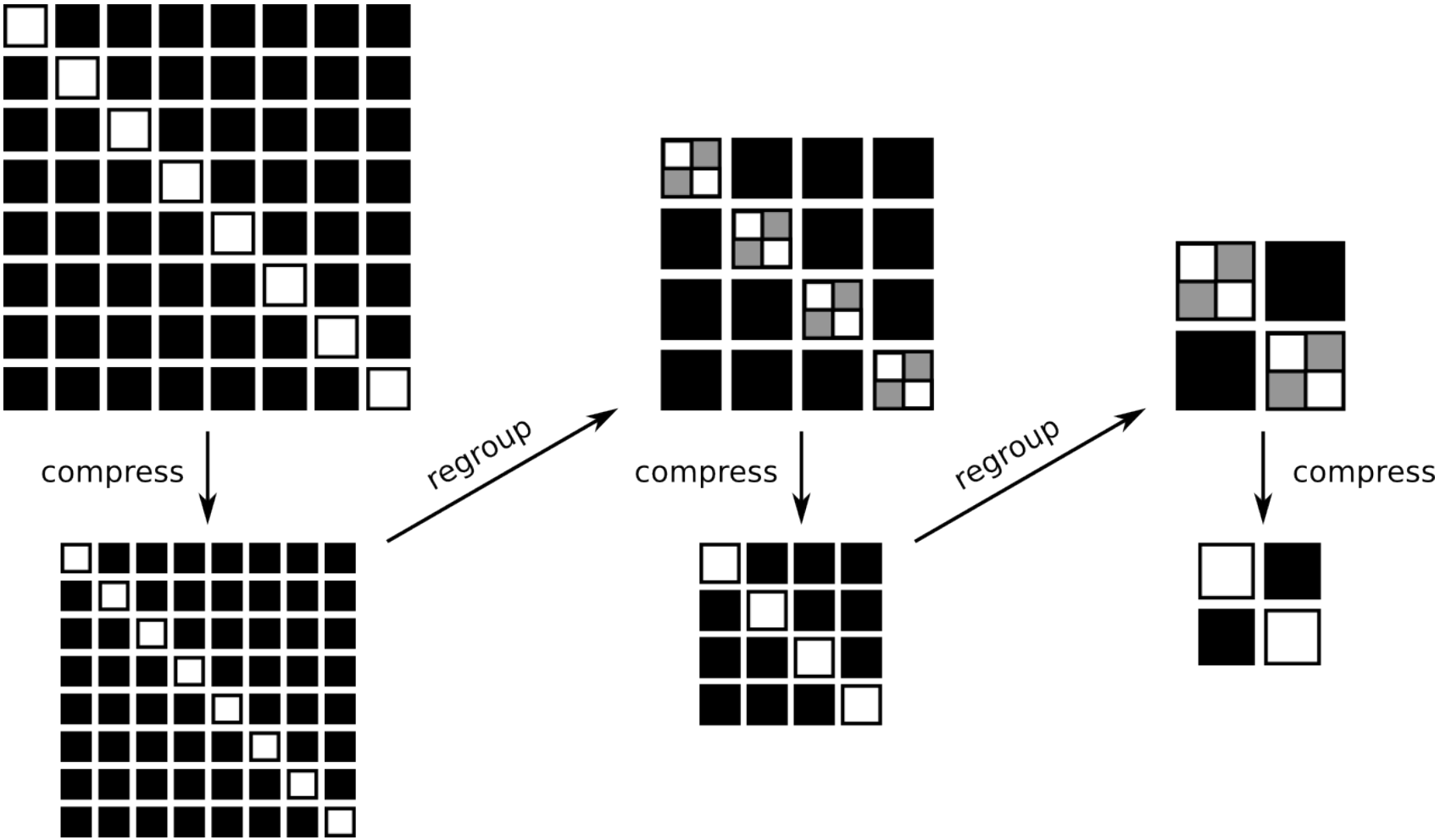
One-level matrix compression

1. Compress the row space of each off-diagonal block row with the ID. Let the L_i be the corresponding row projection matrices.
2. Compress the column space of each off-diagonal block column with the ID. Let the R_j be the corresponding column projection matrices.
3. Approximate the off-diagonal blocks by $A_{ij} \approx L_i S_{ij} R_j$ for $i \neq j$.



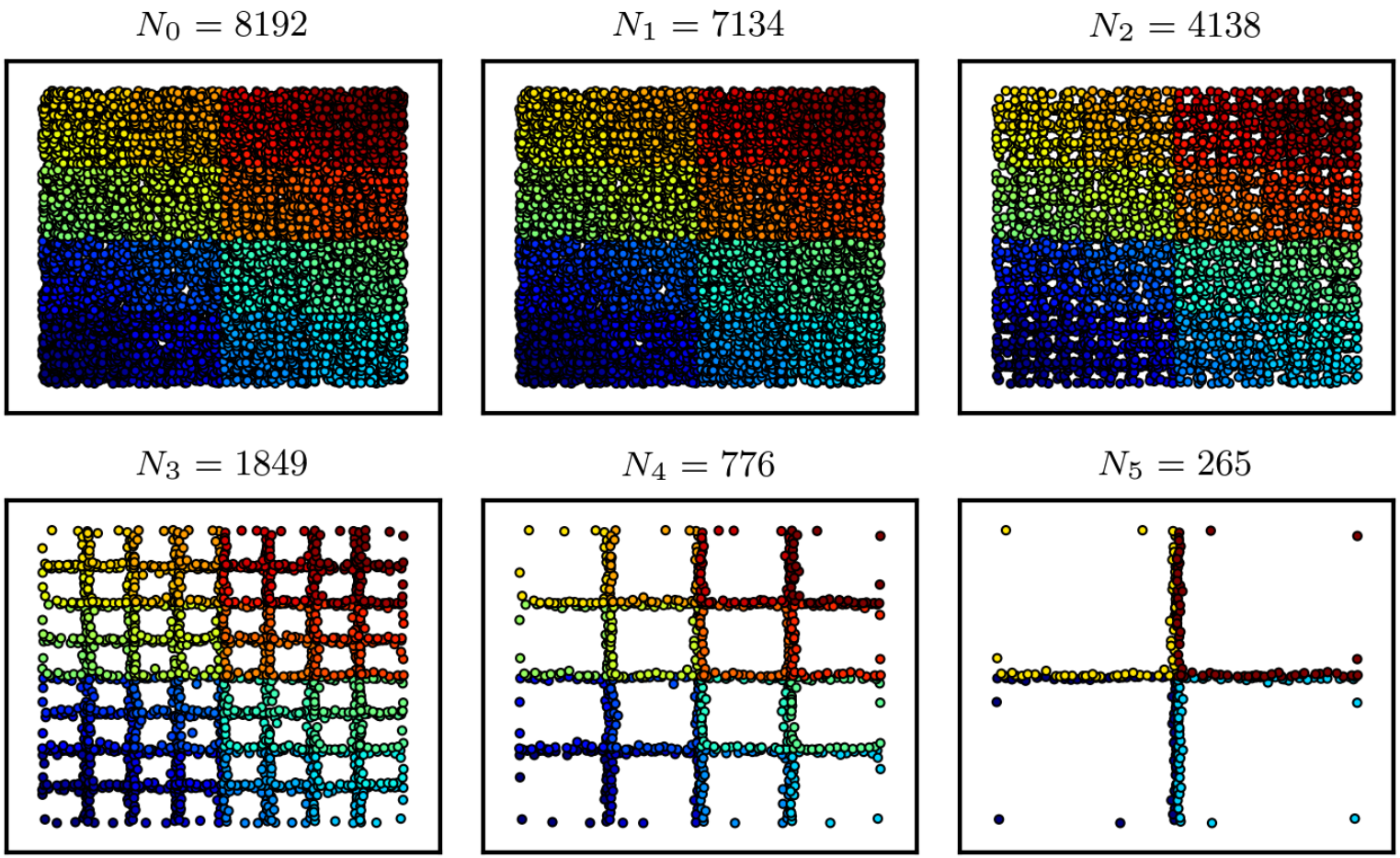
Skeletonization

Multilevel matrix compression



Recursive skeletonization

Data sparsification



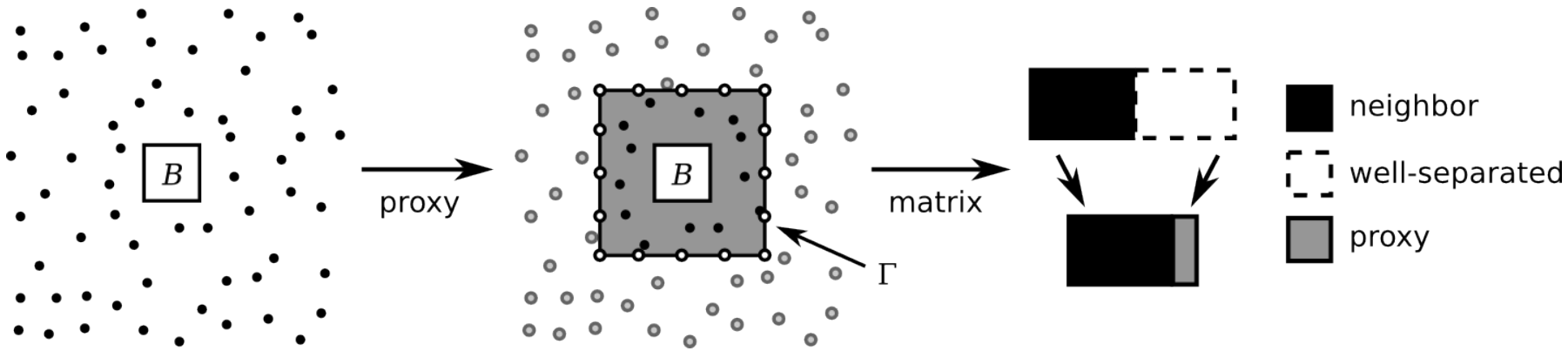
$$G(x,y) = -\frac{1}{2\pi}\log|x-y| \ , \quad \epsilon = 10^{-3}$$

Proxy compression

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

$$u(x) = \int_\Gamma \left[u(y) \frac{\partial G}{\partial \nu(y)}(x,y) - G(x,y) \frac{\partial u}{\partial \nu(y)}(y) \right] ds(y).$$

- Represent well-separated points with a local proxy surface



Compressed matrix representation

Compressed telescoping matrix representation:

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

Compressed matrix representation

Compressed telescoping matrix representation:

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

- Efficient storage (data-sparse)

Storage (2D volume, $\epsilon = 10^{-3}$)

N	Uncompressed	Compressed
8192	537 MB	9.65 MB
131072	137 GB	184 MB

Compressed matrix representation

Compressed telescoping matrix representation:

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

- Efficient storage (data-sparse)
- Fast matrix-vector multiplication (generalized FMM)
- Fast matrix factorization and inverse application

Storage (2D volume, $\epsilon = 10^{-3}$)

N	Uncompressed	Compressed
8192	537 MB	9.65 MB
131072	137 GB	184 MB

Sparse inverse embedding

Recursively expand in sparse form:

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

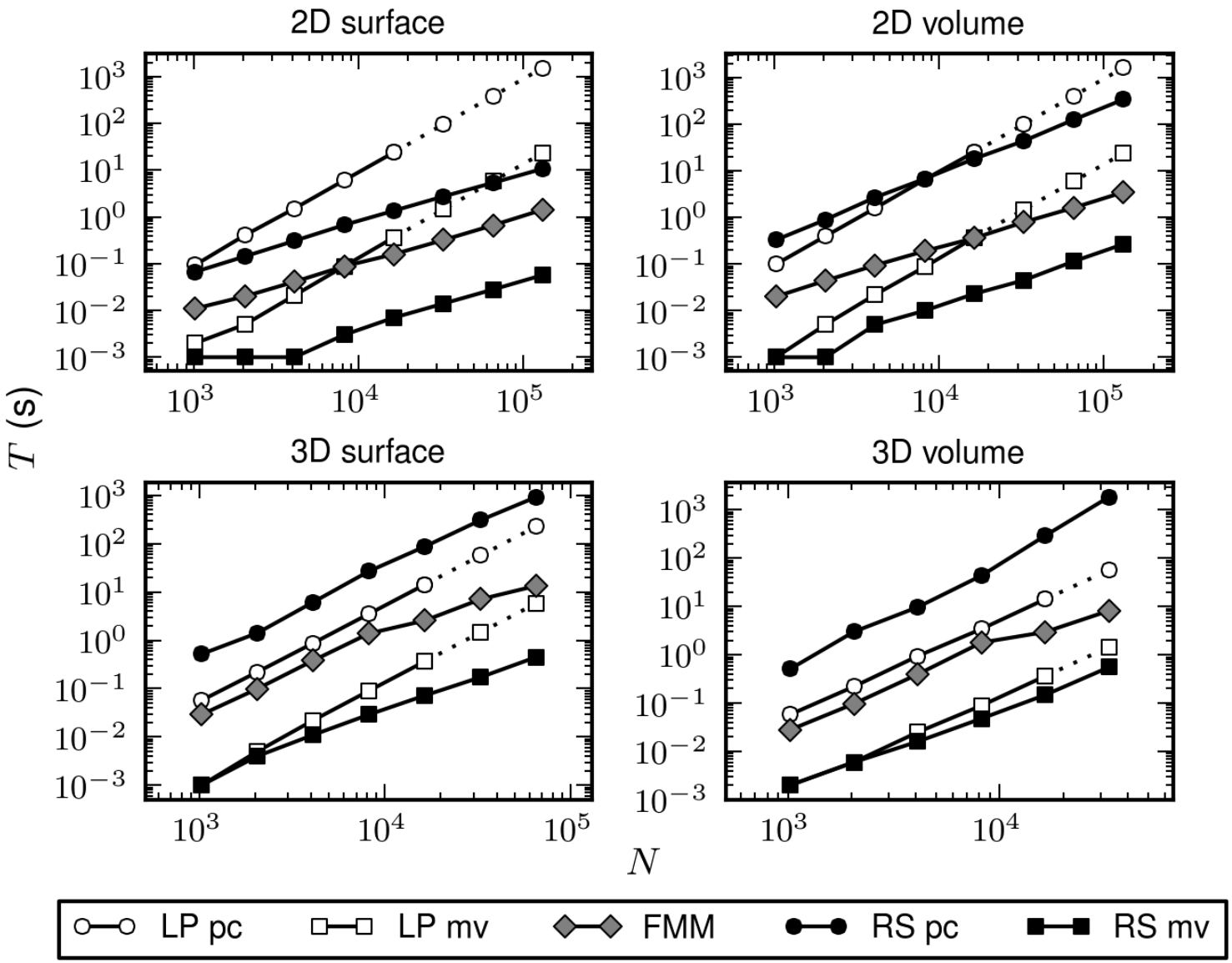
Multilevel inversion formula:

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

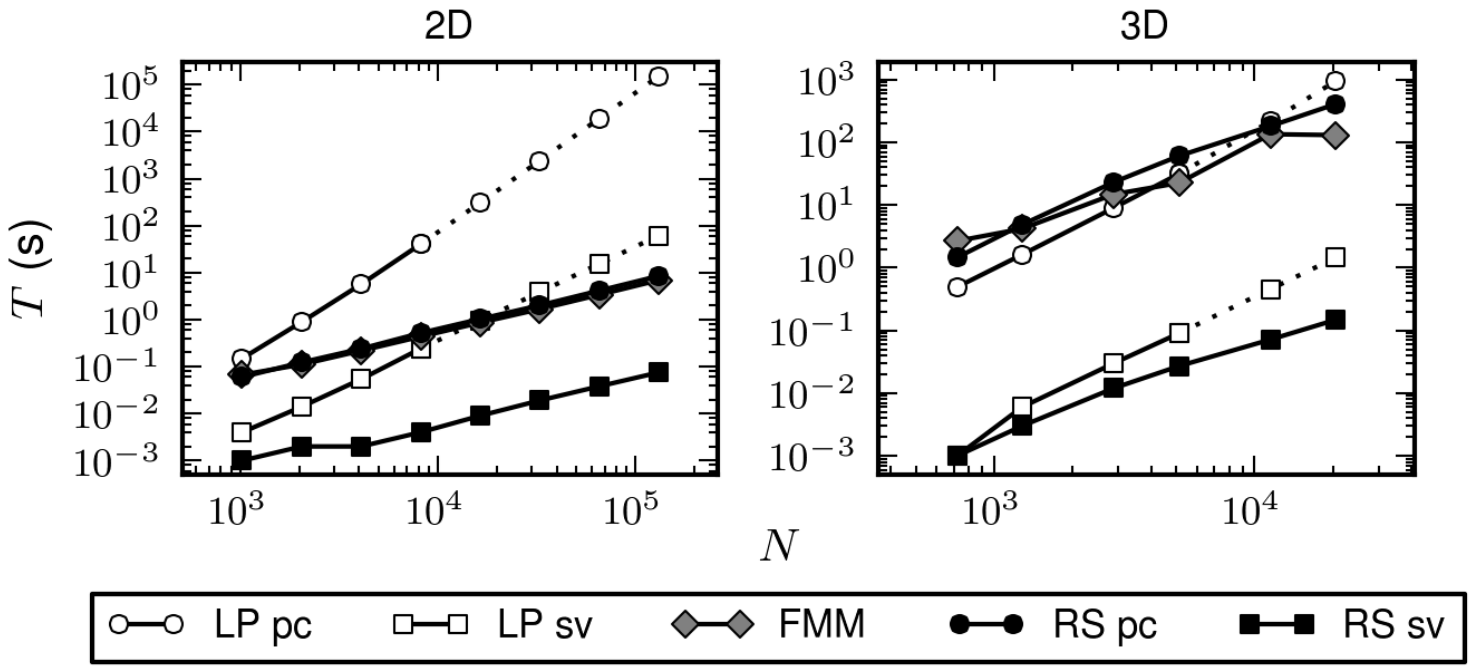
Some comments

- Mild assumptions: low-rank off-diagonal blocks, Green's theorem
- Based on numerical linear algebra rather than analytic expansions
- Kernel-independent (non-oscillatory, elliptic): Laplace, Stokes, Yukawa (screened Poisson), low-frequency Helmholtz, etc.
- Compressed ranks are optimal for the problem at hand
- Trade accuracy for speed: user-specified precision
- Naturally parallelizable via block-sweep structure
- Current limitation: optimal complexity only in 1D (e.g., BIEs in 2D, axisymmetric BIEs in 3D)
- Same ideas can produce an $\mathcal{O}(N)$ compression-based FMM
- Can also work for PDE formulations (Xia et al., 2009)

Laplace FMM


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

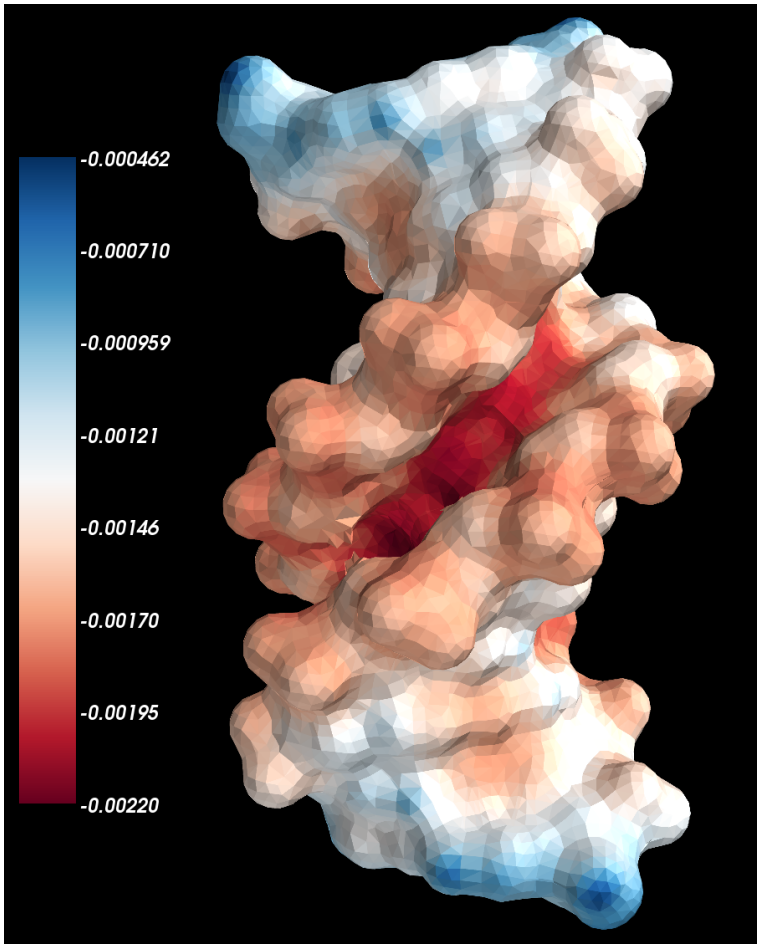
Laplace BIE solver



- Less memory-efficient than FMM/GMRES
- Solve times are 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

Molecular electrostatics

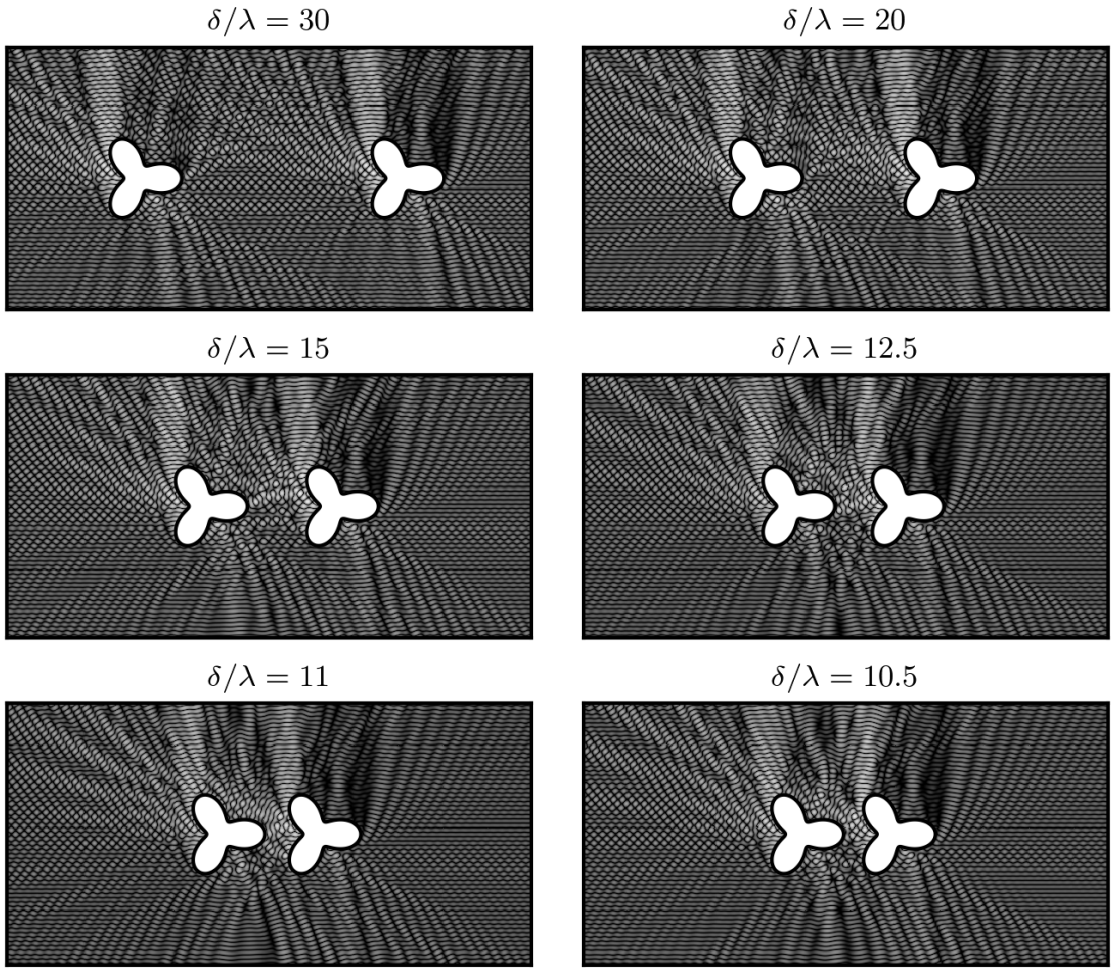


- Piecewise constant dielectric Poisson with interior sources
- $N = 19752$, $\epsilon = 10^{-3}$
- FMM/GMRES: 27 s
- RS precomp: 578 s
- RS solve: 0.08 s
- Break-even: 25 solves

Helmholtz problems

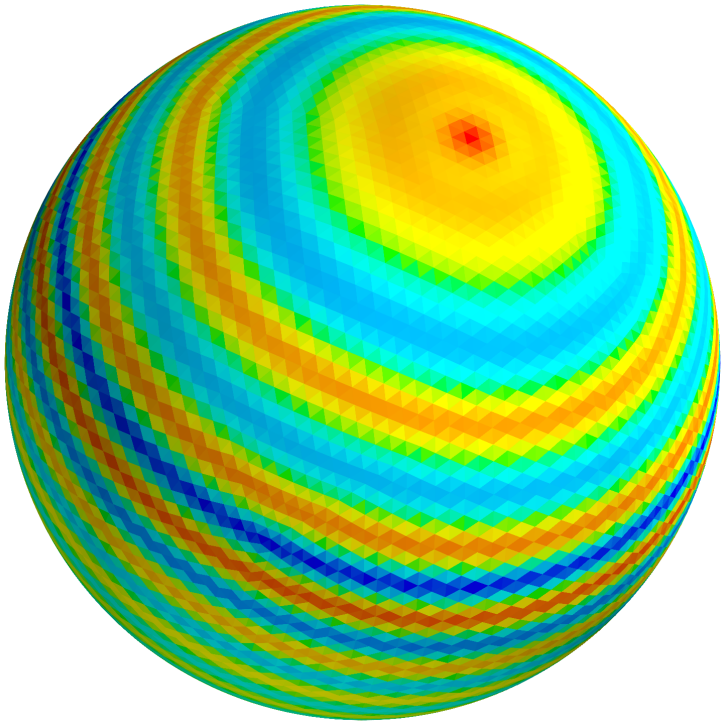
- Interactions are full-rank at high frequency
- Asymptotically no acceleration (becomes $\mathcal{O}(N^3)$ scheme)
- However, remains surprisingly viable: 200λ in 2D, 10λ in 3D
- At low to moderate frequency, sometimes superior to FMM/GMRES due to high iteration counts
- Memory requirements can be a concern
- Effective preconditioners for iterative solvers

Multiple scattering



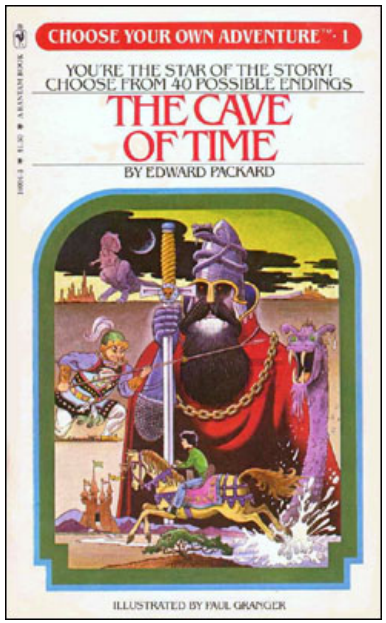
- Each object: 10λ
- Block inverse preconditioner
- Unpreconditioned: 700 iterations
- Preconditioned: 10 iterations
- $50\times$ speedup

Approximate inverse preconditioning



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

Choose your own adventure

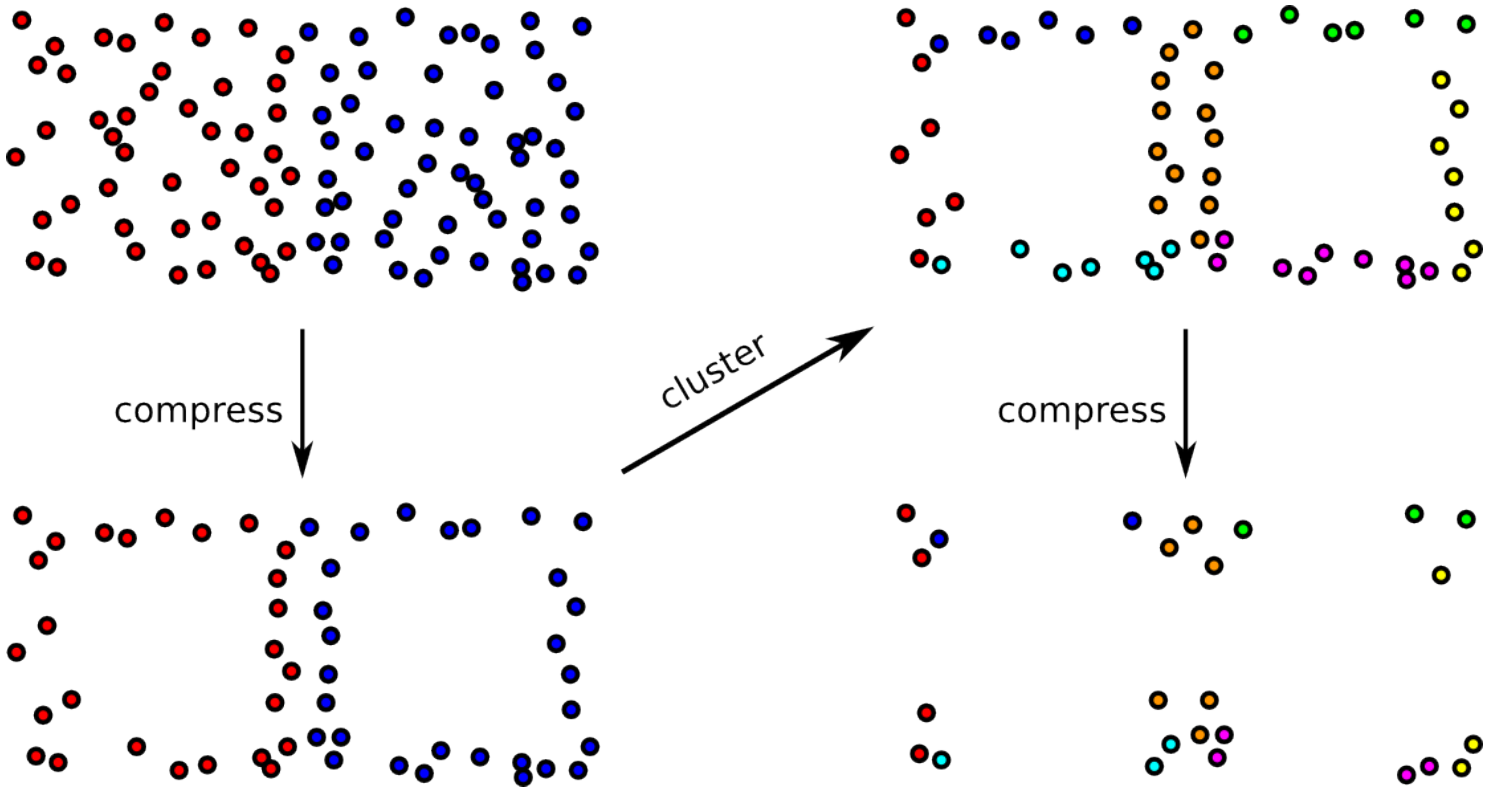


1. Towards a near-optimal direct solver ▸ Go
2. Towards a direct solver for oscillatory kernels ▸ Go
3. Concluding remarks ▸ Go

Towards a near-optimal direct solver

- In 2D and above, the number of skeletons grows with the box size
- Need more compression: skeletonize the skeletons
- Recall that skeletons line up along box boundaries
- Simple idea: cluster according to boundaries and re-compress
- Multiple rounds of compression at each level
- Recurse down on dimensionality: reduce to 1D case
- Expect $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ complexity

Recompression schematic



Return

Towards a direct solver for oscillatory kernels

Previous work:

- BIEs on elongated objects in 2D (Michielssen et al., 1996; Martinsson and Rokhlin, 2007)
- Lippmann-Schwinger in 2D (Chen, 2002)
- BIEs on quasi-planar objects in 3D (Winebrand and Boag, 2009)

No results yet for general BIEs!

Towards a direct solver for oscillatory kernels

Previous work:

- BIEs on elongated objects in 2D (Michielssen et al., 1996; Martinsson and Rokhlin, 2007)
- Lippmann-Schwinger in 2D (Chen, 2002)
- BIEs on quasi-planar objects in 3D (Winebrand and Boag, 2009)

No results yet for general BIEs!

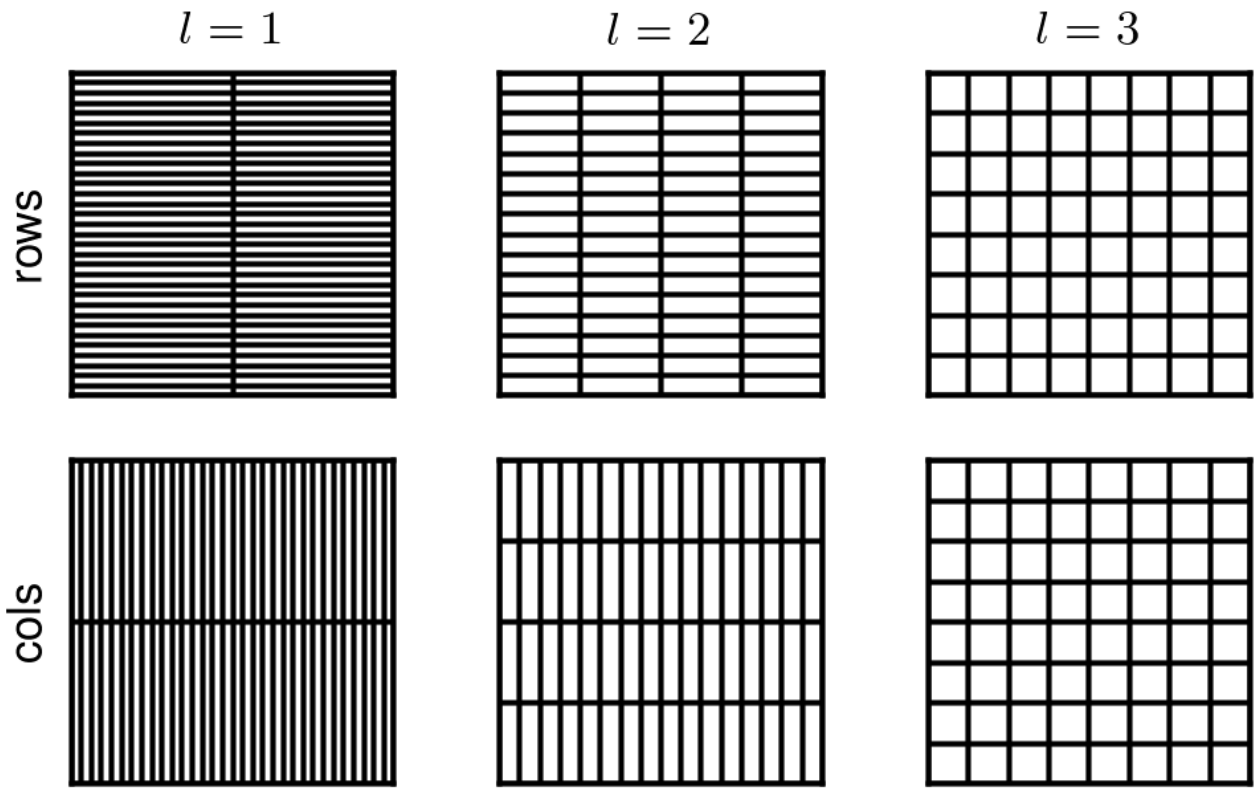
A key observation: at high frequency, the interaction rank between:

- two large boxes is large
- one large box and one small box is small

Exploit with the butterfly algorithm (Michielssen and Boag, 1996).

Butterfly algorithm

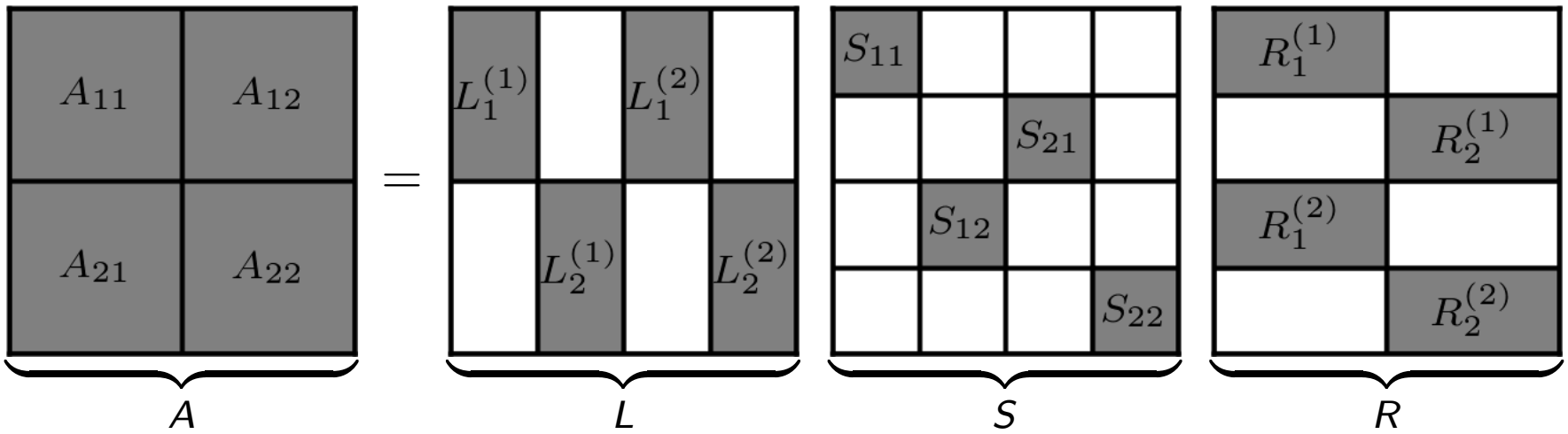
- Blocks of constant "area" have constant rank
- Opposite tree traversal
- Assume no singularities: e.g., consider Fourier integral operators (Candès et al., 2009; Ying, 2009; O'Neil et al., 2010)



Inverting the butterfly

At each level:

$$A = \begin{bmatrix} L_1^{(1)} S_{11} R_1^{(1)} & L_1^{(2)} S_{12} R_2^{(1)} \\ L_2^{(1)} S_{21} R_1^{(2)} & L_2^{(2)} S_{22} R_2^{(2)} \end{bmatrix}$$



Inverse: $A^{-1} = R^{-1} S^{-1} L^{-1}$, or use sparse embedding without D matrix

Butterfly comments

- Multilevel and multidimensional extensions are straightforward
- Preliminary experiments suggest fast inversion
- Expect $\mathcal{O}(N \log N)$ for Fourier integral operators (proxy also available)
- As yet unclear how to treat Helmholtz case (diagonal extraction?)
- Fame, fortune, and glory: revolutionize electromagnetics and imaging

Return

Concluding remarks

- Fast direct solver for non-oscillatory integral equations (Poisson, Stokes, low-frequency Helmholtz, etc.)
- Based on multilevel matrix compression: pure numerical linear algebra
- Following precomputation, solve times are very fast
- Applications: optimization, design, evolution of time-dependent processes in fixed geometries, preconditioning
- Further work: more efficient algorithms, direct solvers for oscillatory kernels, other matrix factorizations