Fast direct solvers for elliptic PDEs

Gunnar Martinsson
The University of Colorado at Boulder

Students:

Adrianna Gillman

Nathan Halko

Sijia Hao

Patrick Young

Collaborators:

Denis Zorin (NYU)

Eduardo Corona (NYU)

Vladimir Rokhlin (Yale)

Mark Tygert (NYU)

The talk will describe "fast direct" techniques for solving the linear systems arising from the discretization of linear boundary value problems (BVPs) of the form

$$\text{(BVP)} \qquad \begin{cases} A\,u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ B\,u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases}$$

where Ω is a domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary Γ, and where A is an elliptic differential operator. Examples include:

- The equations of linear elasticity.
- Stokes' equation.
- Helmholtz' equation (at least at low and intermediate frequencies).
- Time-harmonic Maxwell (at least at low and intermediate frequencies).

Example: Poisson equation with Dirichlet boundary data:

$$\begin{cases} -\Delta\,u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \quad u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases}$$

Discretization of linear Boundary Value Problems

↙ ↘

Direct discretization of the differential operator via Finite Elements, Finite Differences, ...

↓

$N \times N$ discrete linear system.
Very large, sparse, ill-conditioned.

↓

Fast solvers:
iterative (multigrid), $O(N)$,
direct (nested dissection), $O(N^{3/2})$.

Conversion of the BVP to a Boundary Integral Operator (BIE).

↓

Discretization of (BIE) using Nyström, collocation, BEM, ...

↓

$N \times N$ discrete linear system.
Moderate size, dense,
(often) well-conditioned.

↓

Iterative solver accelerated by fast matrix-vector multiplier, $O(N)$.

Discretization of linear Boundary Value Problems

Direct discretization of the differential operator via Finite Elements, Finite Differences, ...

↓

$N \times N$ discrete linear system.
Very large, sparse, ill-conditioned.

↓

Fast solvers:
iterative (multigrid), $O(N)$,
direct (nested dissection), $O(N^{3/2})$.
$O(N)$ direct solvers.

Conversion of the BVP to a Boundary Integral Operator (BIE).

↓

Discretization of (BIE) using Nyström, collocation, BEM, ...

↓

$N \times N$ discrete linear system.
Moderate size, dense,
(often) well-conditioned.

↓

Iterative solver accelerated by fast matrix-vector multiplier, $O(N)$.
$O(N)$ direct solvers.

What does a "direct" solver mean in this context?

Basically, it is a solver that is not "iterative" ...

Given a computational tolerance ε, and a linear system

$$\mathbf{A}\,\mathbf{u} = \mathbf{b}, \tag{2}$$

(where the system matrix $\mathbf{A}$ is often defined implicitly), a *direct solver* constructs an operator $\mathbf{T}$ such that

$$||\mathbf{A}^{-1} - \mathbf{T}|| \leq \varepsilon.$$

Then an approximate solution to (2) is obtained by simply evaluating

$$\mathbf{u}_{\text{approx}} = \mathbf{T}\,\mathbf{b}.$$

The matrix $\mathbf{T}$ is typically constructed in a *compressed format* that allows the matrix-vector product $\mathbf{T}\,\mathbf{b}$ to be evaluated rapidly.

Variation: Find factors $\mathbf{B}$ and $\mathbf{C}$ such that $||\mathbf{A} - \mathbf{B}\,\mathbf{C}|| \leq \varepsilon$, and linear solves involving the matrices $\mathbf{B}$ and $\mathbf{C}$ are fast. (LU-decomposition, Cholesky, *etc.*)

"Iterative" versus "direct" solvers

Two classes of methods for solving an $N \times N$ linear algebraic system

$$\mathbf{A}\,\mathbf{u} = \mathbf{b}.$$

Iterative methods:

Examples: GMRES, conjugate gradients, Gauss-Seidel, *etc.*

Construct a sequence of vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots$ that (hopefully!) converge to the exact solution.

Many iterative methods access $\mathbf{A}$ only via its action on vectors.

Often require problem specific preconditioners.

High performance when they work well.
$O(N)$ solvers.

Direct methods:

Examples: Gaussian elimination,
LU factorizations, matrix inversion, *etc.*

Always give an answer. Deterministic.

Robust. No convergence analysis.

Great for multiple right hand sides.

Have often been considered too slow for high performance computing.

(Directly access elements or blocks of $\mathbf{A}$.)

(Exact except for rounding errors.)

Advantages of direct solvers over iterative solvers:

1. Applications that require a very large number of solves:
   - Molecular dynamics.
   - Scattering problems.
   - Optimal design. (Local updates to the system matrix are cheap.)

   A couple of orders of magnitude speed-up is often possible.

2. Problems that are relatively ill-conditioned:
   - Scattering problems near resonant frequencies.
   - Ill-conditioning due to geometry (elongated domains, percolation, etc).
   - Ill-conditioning due to lazy handling of corners, cusps, *etc.*
   - Finite element and finite difference discretizations.

   Scattering problems intractable to existing methods can (sometimes) be solved.

3. Direct solvers can be adapted to construct spectral decompositions:
   - Analysis of vibrating structures. Acoustics.
   - Buckling of mechanical structures.
   - Wave guides, bandgap materials, *etc.*

Advantages of direct solvers over iterative solvers, continued:

Perhaps most important: **Engineering considerations.**

Direct methods tend to be more robust than iterative ones.

This makes them more suitable for "black-box" implementations.

Commercial software developers appear to avoid implementing iterative solvers whenever possible. (Sometimes for good reasons.)

The effort to develop direct solvers aims to help in the development of general purpose software packages solving the basic linear boundary value problems of mathematical physics.

Fast direct solvers for elliptic PDEs in the literature:

(Apologies to co-workers: A. Gillman, L. Greengard, D. Gueyffier, V. Rokhlin, M. Tygert, P. Young, ...)

1991 Data-sparse matrix algebra / wavelets, *Beylkin, Coifman, Rokhlin, et al*

1993 Fast inversion of 1D operators *V. Rokhlin and P. Starr*

1996 scattering problems, *E. Michielssen, A. Boag and W.C. Chew*,

1998 factorization of non-standard forms, *G. Beylkin, J. Dunn, D. Gines*,

1998 $\mathcal{H}$-matrix methods, *W. Hackbusch, B. Khoromskijet, S. Sauter,...*,

2000 Cross approximation, matrix skeletons, etc., E. Tyrtyshnikov.

2002 $O(N^{3/2})$ inversion of Lippmann-Schwinger equations, *Y. Chen*,

2002 "Hierarchically Semi-Separable" matrices, *M. Gu, S. Chandrasekharan.*

2002 (1999?) $\mathcal{H}^2$-matrix methods, *S. Börm, W. Hackbusch, B. Khoromskijet, S. Sauter.*

2004 Inversion of "FMM structure," *S. Chandrasekharan, T. Pals.*

2004 Proofs of compressibility, *M. Bebendorf, S. Börm, W. Hackbusch, ...*

2007 Accelerated nested diss. via HSS, *S. Chandrasekharan, M. Gu, X.S. Li, J. Xia.*

2008 Accelerated nested diss. via $\mathcal{H}$-mats, *L. Grasedyck, R. Kriemann, S. LeBorne.*

2010 construction of $\mathbf{A}^{-1}$ via randomized sampling, *L. Lin, J. Lu, L. Ying.*

Current work: A. Barnett, J. Bremer, E. Michielsen, V. Rokhlin, M. Tygert, ...

Current status — problems with non-oscillatory kernels (Laplace, elasticity, Stokes, *etc*).

Problems on 1D domains:

- *Integral equations on the line:* Done. $O(N)$ with very small constants.
- *Boundary Integral Equations in $\mathbb{R}^2$:* Done. $O(N)$ with small constants.
- *BIEs on axisymmetric surfaces in $\mathbb{R}^3$:* Done. $O(N)$ with small constants.

Problems on 2D domains:

- *"FEM" matrices for elliptic PDEs in the plane:* $O(N)$ algorithms exist. Work remains.
- *Volume Int. Eq. in the plane (e.g. low frequency Lippman-Schwinger):* $O(N\,(\log N)^p)$ algorithms exist. $O(N)$ and high accuracy methods are under development.
- *Boundary Integral Equations in $\mathbb{R}^3$:* $O(N\,(\log N)^p)$ algorithms exist. $O(N)$ and high accuracy methods are under development.

Problems on 3D domains:

- *"FEM" matrices for elliptic PDEs:* Very active area!
  (Grasedyck & LeBorne; Michielssen; Xia; Ying; . . .)
- *Volume Int. Eq.:* Memory constraints currently seem problematic.

Current status — problems with oscillatory kernels (Helmholtz, Maxwell, etc.).
*Direct solvers are **extremely** desirable in this environment!*

Problems on 1D domains:

- *Integral equations on the line:* Done — $O(N)$ with small constants.
- *Boundary Integral Equations in $\mathbb{R}^2$:* ???
- *("Elongated" surfaces in $\mathbb{R}^2$ and $\mathbb{R}^3$:* Done — $O(N \log N)$.)

Problems on 2D domains:

- *"FEM" matrices for Helmholtz equation in the plane:* ???
  ($O(N^{1.5})$ inversion is possible.)
- *Volume Int. Eq. in the plane (e.g. high frequency Lippman-Schwinger):* ???
- *Boundary Integral Equations in $\mathbb{R}^3$:* ???

Problems on 3D domains:

- ???? ($O(N^2)$ inversion sometimes possible — memory requirement is a concern.)

Recent work by B. Engquist and L. Ying — very efficient pre-conditioners based on structured matrix calculations. "Semi-direct."

Direct solvers based on Hierarchically Semi-Separable matrices

Consider a linear system

$$\mathbf{A}\,\mathbf{q} = \mathbf{f},$$

where $\mathbf{A}$ is a "block-separable" matrix consisting of $p \times p$ blocks of size $n \times n$:

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} \\ \mathbf{A}_{21} & \mathbf{D}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{D}_{33} & \mathbf{A}_{34} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{D}_{44} \end{bmatrix}. \qquad \text{(Shown for } p = 4.\text{)}$$

Core assumption: Each off-diagonal block $\mathbf{A}_{ij}$ admits the factorization

$$\begin{array}{ccccc} \mathbf{A}_{ij} & = & \mathbf{U}_i & \tilde{\mathbf{A}}_{ij} & \mathbf{V}_j^* \\ n \times n & & n \times k & k \times k & k \times n \end{array}$$

where the rank k is significantly smaller than the block size n. (Say $k \approx n/2$.)

The critical part of the assumption is that all off-diagonal blocks in the i'th row use the same basis matrices $\mathbf{U}_i$ for their column spaces (and analogously all blocks in the j'th column use the same basis matrices $\mathbf{V}_j$ for their row spaces).

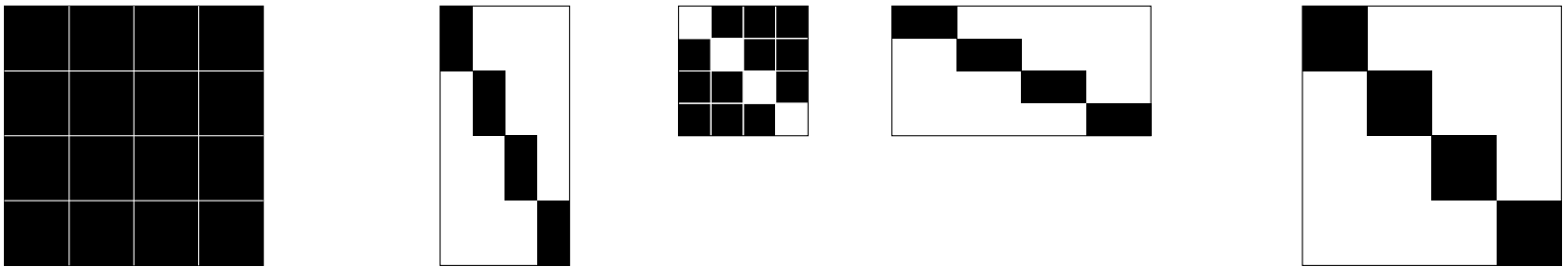
We get $\mathbf{A} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{U}_1 \tilde{\mathbf{A}}_{12} \mathbf{V}_2^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{13} \mathbf{V}_3^* & \mathbf{U}_1 \tilde{\mathbf{A}}_{14} \mathbf{V}_4^* \\ \mathbf{U}_2 \tilde{\mathbf{A}}_{21} \mathbf{V}_1^* & \mathbf{D}_{22} & \mathbf{U}_2 \tilde{\mathbf{A}}_{23} \mathbf{V}_3^* & \mathbf{U}_2 \tilde{\mathbf{A}}_{24} \mathbf{V}_4^* \\ \mathbf{U}_3 \tilde{\mathbf{A}}_{31} \mathbf{V}_1^* & \mathbf{U}_3 \tilde{\mathbf{A}}_{32} \mathbf{V}_2^* & \mathbf{D}_{33} & \mathbf{U}_3 \tilde{\mathbf{A}}_{34} \mathbf{V}_4^* \\ \mathbf{U}_4 \tilde{\mathbf{A}}_{41} \mathbf{V}_1^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{42} \mathbf{V}_2^* & \mathbf{U}_4 \tilde{\mathbf{A}}_{43} \mathbf{V}_3^* & \mathbf{D}_{44} \end{bmatrix}$.

Then $\mathbf{A}$ admits the factorization:

$$\mathbf{A} = \underbrace{\begin{bmatrix} \mathbf{U}_1 & & & \\ & \mathbf{U}_2 & & \\ & & \mathbf{U}_3 & \\ & & & \mathbf{U}_4 \end{bmatrix}}_{=\mathbf{U}} \underbrace{\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{12} & \tilde{\mathbf{A}}_{13} & \tilde{\mathbf{A}}_{14} \\ \tilde{\mathbf{A}}_{21} & \mathbf{0} & \tilde{\mathbf{A}}_{23} & \tilde{\mathbf{A}}_{24} \\ \tilde{\mathbf{A}}_{31} & \tilde{\mathbf{A}}_{32} & \mathbf{0} & \tilde{\mathbf{A}}_{34} \\ \tilde{\mathbf{A}}_{41} & \tilde{\mathbf{A}}_{42} & \tilde{\mathbf{A}}_{43} & \mathbf{0} \end{bmatrix}}_{=\tilde{\mathbf{A}}} \underbrace{\begin{bmatrix} \mathbf{V}_1^* & & & \\ & \mathbf{V}_2^* & & \\ & & \mathbf{V}_3^* & \\ & & & \mathbf{V}_4^* \end{bmatrix}}_{=\mathbf{V}^*} + \underbrace{\begin{bmatrix} \mathbf{D}_1 & & & \\ & \mathbf{D}_2 & & \\ & & \mathbf{D}_3 & \\ & & & \mathbf{D}_4 \end{bmatrix}}_{=\mathbf{D}}$$

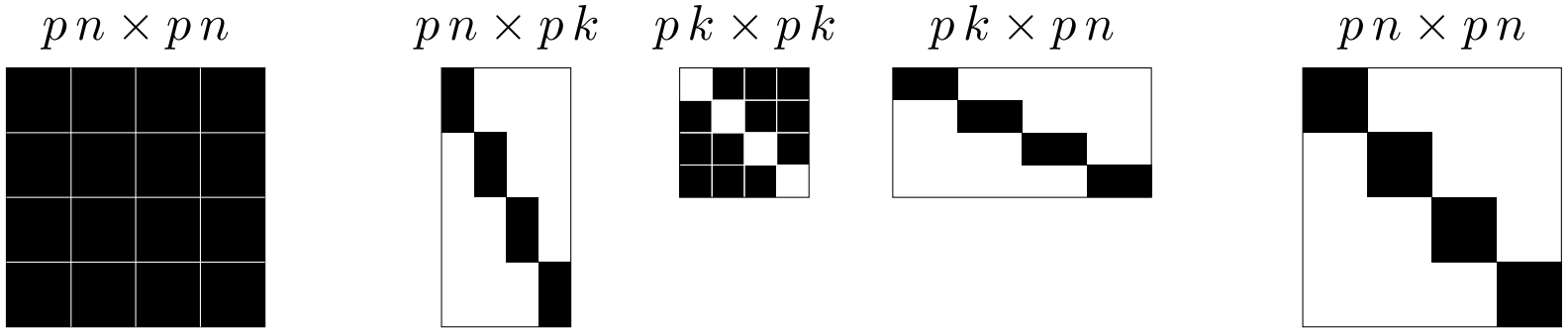
or

$$\underset{pn \times pn}{\mathbf{A}} = \underset{pn \times pk}{\mathbf{U}} \; \underset{pk \times pk}{\tilde{\mathbf{A}}} \; \underset{pk \times pn}{\mathbf{V}^*} + \underset{pn \times pn}{\mathbf{D}},$$



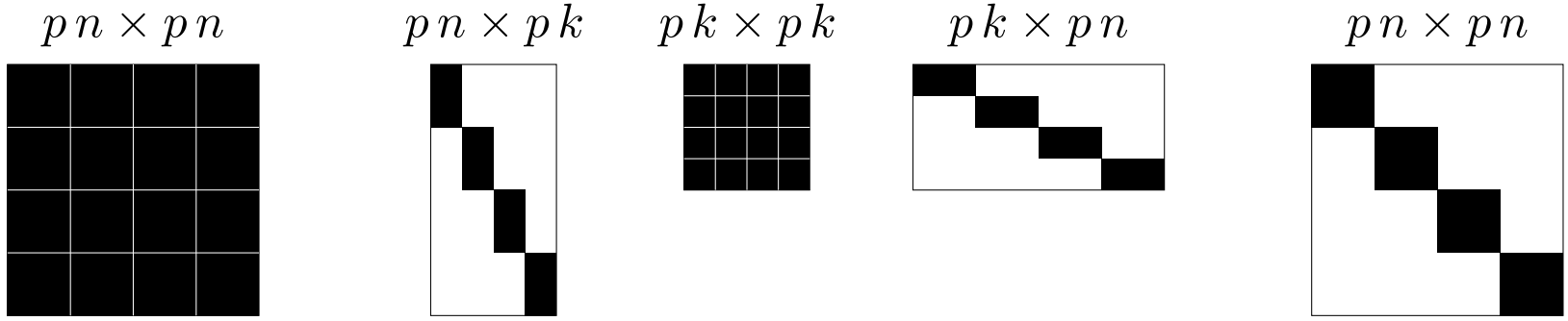
Lemma: [Variation of Woodbury] If an $N \times N$ matrix $\mathbf{A}$ admits the factorization

$$\mathbf{A} = \mathbf{U}\ \tilde{\mathbf{A}}\ \mathbf{V}^* + \mathbf{D},$$



then

$$\mathbf{A}^{-1} = \mathbf{E}\ (\tilde{\mathbf{A}} + \hat{\mathbf{D}})^{-1}\ \mathbf{F}^* + \mathbf{G},$$



where (provided all intermediate matrices are invertible)

$$\hat{\mathbf{D}} = \left(\mathbf{V}^*\,\mathbf{D}^{-1}\,\mathbf{U}\right)^{-1}, \quad \mathbf{E} = \mathbf{D}^{-1}\,\mathbf{U}\,\hat{\mathbf{D}}, \quad \mathbf{F} = (\hat{\mathbf{D}}\,\mathbf{V}^*\,\mathbf{D}^{-1})^*, \quad \mathbf{G} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\,\mathbf{U}\,\hat{\mathbf{D}}\,\mathbf{V}^*\,\mathbf{D}^{-1}.$$

Note: All matrices set in blue are block diagonal.

The Woodbury formula replaces the task of inverting a $p\,n \times p\,n$ matrix by the task of inverting a $p\,k \times p\,k$ matrix.

The cost is reduced from $(p\,n)^3$ to $(p\,k)^3$.

We do not yet have a "fast" scheme ...

(Recall: $\mathbf{A}$ has $p \times p$ blocks, each of size $n \times n$ and of rank k.)

We must recurse!

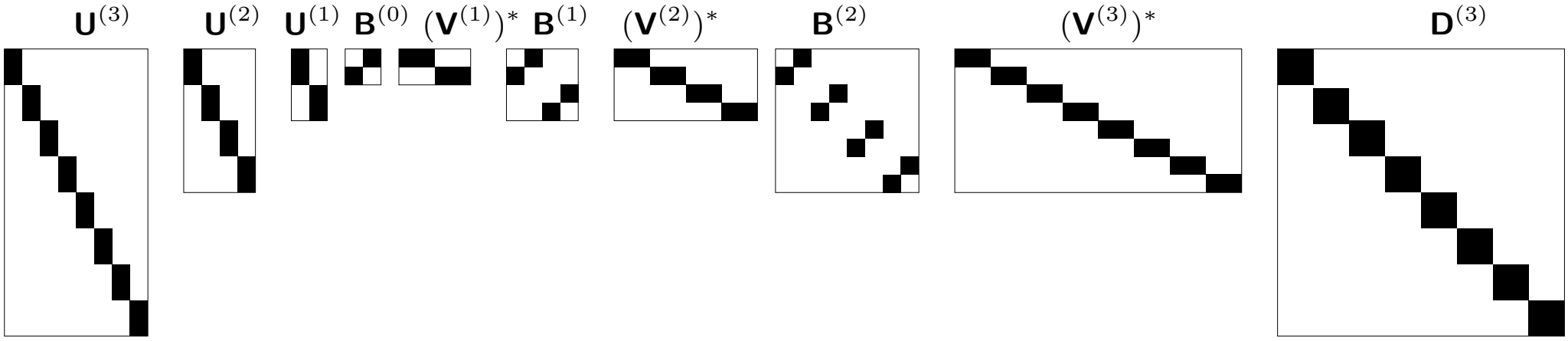
Using a telescoping factorization of $\mathbf{A}$ (a "hierarchically block-separable" representation):

$$\mathbf{A} = \mathbf{U}^{(3)}\left(\mathbf{U}^{(2)}\left(\mathbf{U}^{(1)}\,\mathbf{B}^{(0)}\,\mathbf{V}^{(1)}\right)^* + \mathbf{B}^{(1)}\right)\left(\mathbf{V}^{(2)}\right)^* + \mathbf{B}^{(2)}\Big)\left(\mathbf{V}^{(3)}\right)^* + \mathbf{D}^{(3)},$$

we have a formula

$$\mathbf{A}^{-1} = \mathbf{E}^{(3)}\left(\mathbf{E}^{(2)}\left(\mathbf{E}^{(1)}\,\hat{\mathbf{D}}^{(0)}\,\mathbf{F}^{(1)}\right)^* + \hat{\mathbf{D}}^{(1)}\right)\left(\mathbf{F}^{(2)}\right)^* + \hat{\mathbf{D}}^{(2)}\Big)\left(\mathbf{V}^{(3)}\right)^* + \hat{\mathbf{D}}^{(3)}.$$

Block structure of factorization:

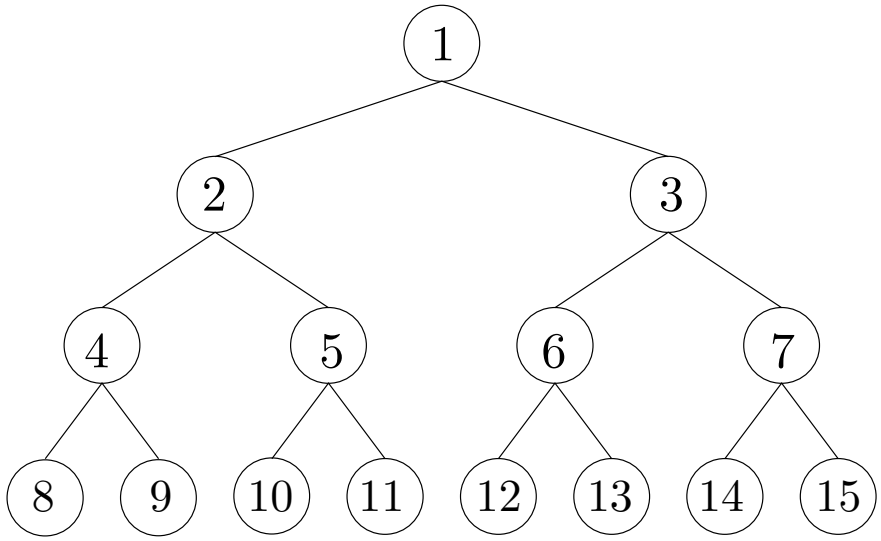


All matrices are now block diagonal except $\hat{\mathbf{D}}^{(0)}$*, which is small.*

Formal definition of an HSS matrix

Suppose $\mathcal{T}$ is a binary tree.

For a node τ in the tree, let I_τ denote the corresponding index vector.



Numbering of nodes in a fully populated binary tree with $L = 3$ levels.
The root is the original index vector $I = I_1 = [1, 2, \ldots, 400]$.

Formal definition of an HSS matrix

Suppose $\mathcal{T}$ is a binary tree.

For a node τ in the tree, let I_τ denote the corresponding index vector.

For leaves σ and τ, set $\mathbf{A}_{\sigma,\tau} = \mathbf{A}(I_\sigma, I_\tau)$ and suppose that all off-diagonal blocks satisfy

$$\begin{array}{ccccccc} \mathbf{A}_{\sigma,\tau} & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^* & & \sigma \neq \tau \\ n \times n & & n \times k & k \times k & k \times n & & \end{array}$$

For non-leaves σ and τ, let $\{\sigma_1, \sigma_2\}$ denote the children of σ, and let $\{\tau_1, \tau_2\}$ denote the children of τ. Set

$$\mathbf{A}_{\sigma,\tau} = \begin{bmatrix} \tilde{\mathbf{A}}_{\sigma_1,\tau_1} & \tilde{\mathbf{A}}_{\sigma_1,\tau_2} \\ \tilde{\mathbf{A}}_{\sigma_2,\tau_1} & \tilde{\mathbf{A}}_{\sigma_2,\tau_2} \end{bmatrix}$$

Then suppose that the off-diagonal blocks satisfy

$$\begin{array}{ccccccc} \mathbf{A}_{\sigma,\tau} & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^* & & \sigma \neq \tau \\ 2k \times 2k & & 2k \times k & k \times k & k \times 2k & & \end{array}$$

	Name:	Size:	Function:
For each leaf	$\mathbf{D}_\tau$	$n \times n$	The diagonal block $\mathbf{A}(I_\tau, I_\tau)$.
node τ:	$\mathbf{U}_\tau$	$n \times k$	Basis for the columns in the blocks in row τ.
	$\mathbf{V}_\tau$	$n \times k$	Basis for the rows in the blocks in column τ.
For each parent	$\mathbf{B}_\tau$	$2k \times 2k$	Interactions between the children of τ.
node τ:	$\mathbf{U}_\tau$	$2k \times k$	Basis for the columns in the (reduced) blocks in row τ.
	$\mathbf{V}_\tau$	$2k \times k$	Basis for the rows in the (reduced) blocks in column τ.

An HSS matrix $\mathbf{A}$ *associated with a tree* $\mathcal{T}$ *is fully specified if the factors listed above are provided.*

Choice of basis matrices (our approach is non-standard):

Recall: The HSS structure relies on factorizations such as (for $k < n$)

$$\begin{array}{ccccc} \mathbf{A}_{\sigma,\tau} & = & \mathbf{U}_\sigma & \tilde{\mathbf{A}}_{\sigma,\tau} & \mathbf{V}_\tau^* \\ n \times n & & n \times k & k \times k & k \times n \end{array}$$

For HSS matrix algebra to be numerically stable, it is critical that the basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ be well-conditioned.

The gold-standard is to have $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ be orthonormal (i.e. $\sigma_j(\mathbf{U}_\tau) = \sigma_j(\mathbf{V}_\tau) = 1$ for $j = 1, 2, \ldots, k$), and this is commonly enforced.

We have decided to instead use *interpolatory decompositions* in which:

1. $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ each contain the $k \times k$ identity matrix as a submatrix.
2. $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ are "reasonably" well-conditioned.
3. $\tilde{\mathbf{A}}_{\sigma,\tau}$ is a submatrix of $\mathbf{A}$ for all σ, τ.

Our choice leads to some loss of accuracy, but vastly simplifies the task of computing compressed representations in the context of integral equations. (For instance, if the original $\mathbf{A}$ represents a Nyström discretization, then the HSS representation on each level is also a Nyström discretization, only with modified diagonal blocks, and on coarser discretizations.)

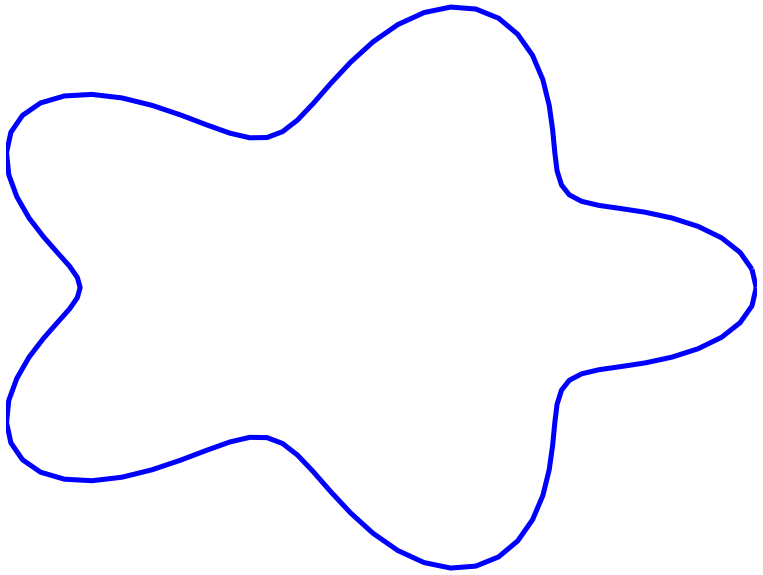
Sample environment: Discretization of 1D integral operators

For simplicity, consider a 100×100 matrix $\mathbf{A}$ approximating the operator

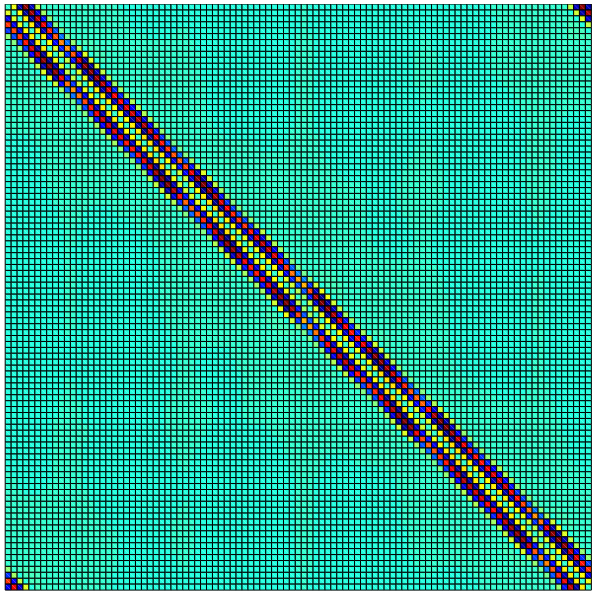
$$[\mathcal{S}_\Gamma\, u](\mathbf{x}) = u(\mathbf{x}) + \int_\Gamma \log|\mathbf{x} - \mathbf{y}|\, u(\mathbf{y})\, ds(\mathbf{y}).$$

The matrix $\mathbf{A}$ is characterized by:

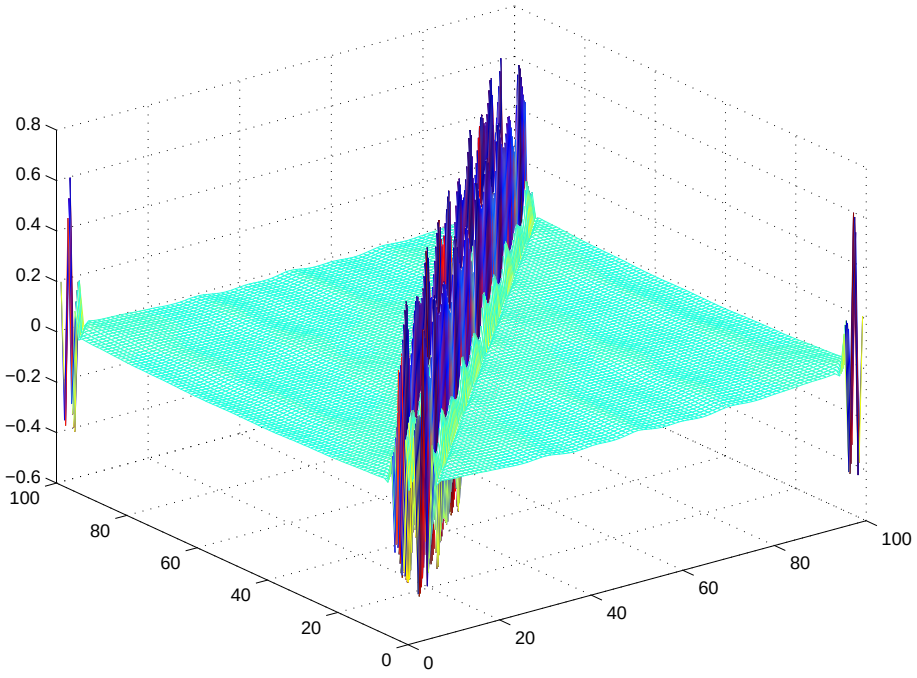
- Irregular behavior near the diagonal.
- Smooth entries away from the diagonal.



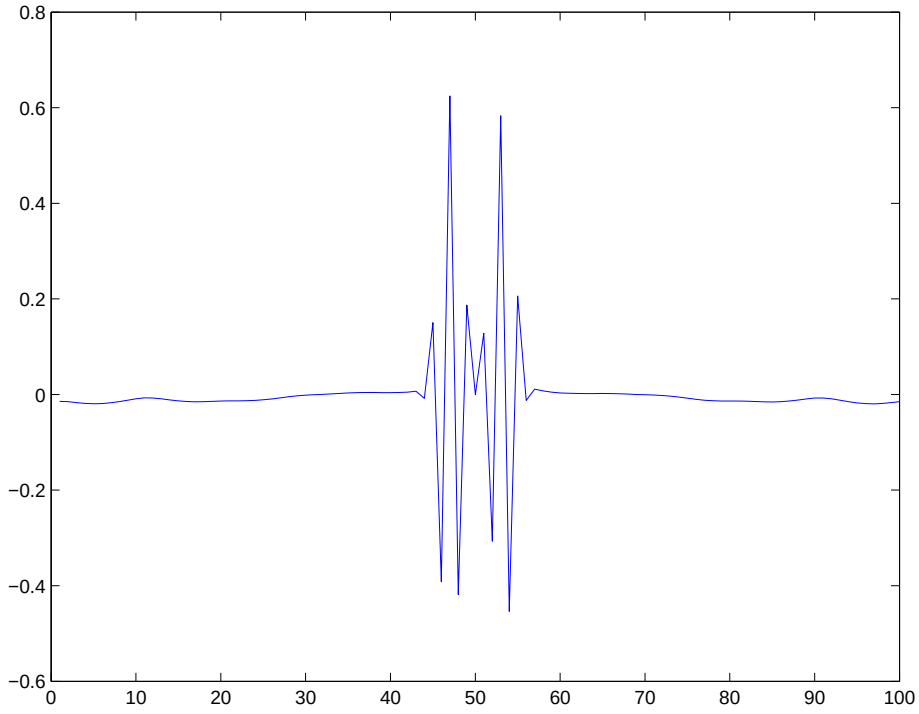
The contour Γ.



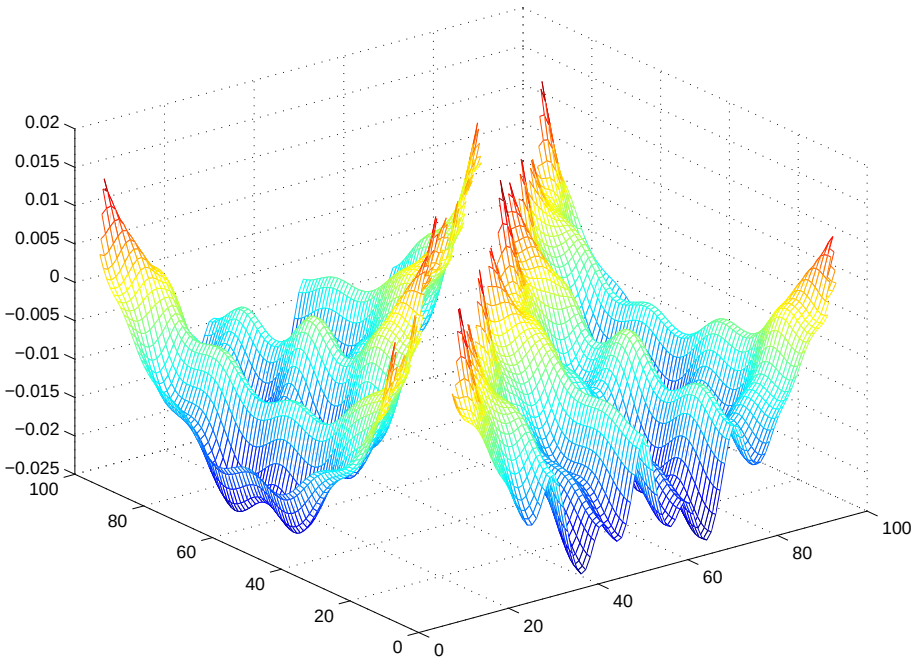
The matrix $\mathbf{A}$.



Plot of a_{ij} vs i and j

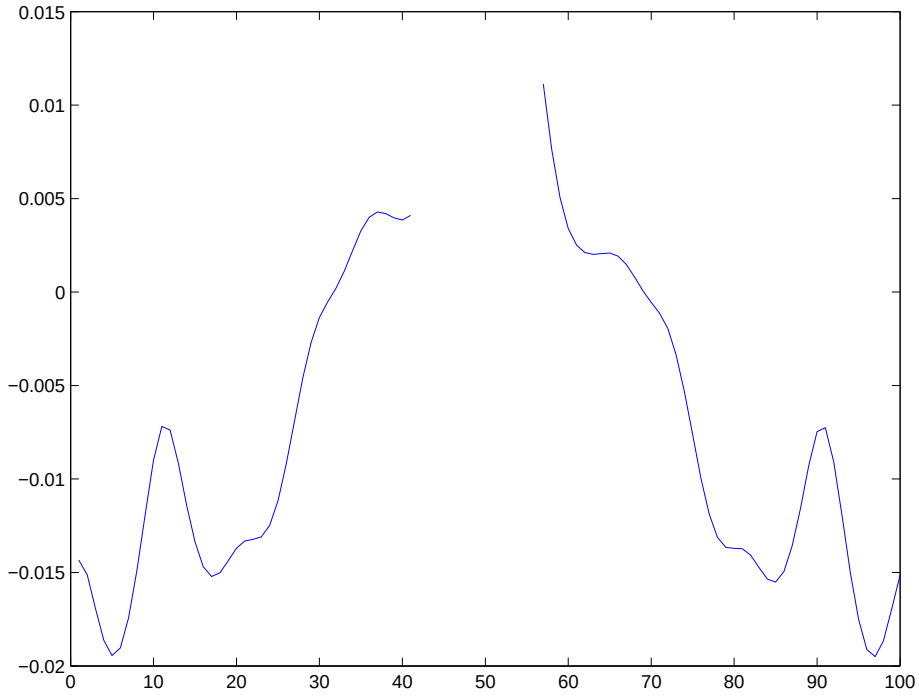


The 50^{th} row of **A**



Plot of a_{ij} vs i and j

(without the diagonal entries)

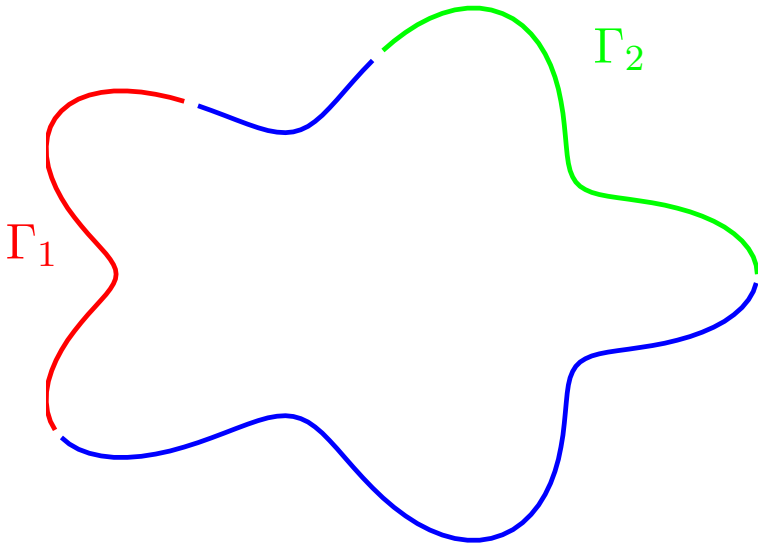


The 50^{th} row of $\mathbf{A}$

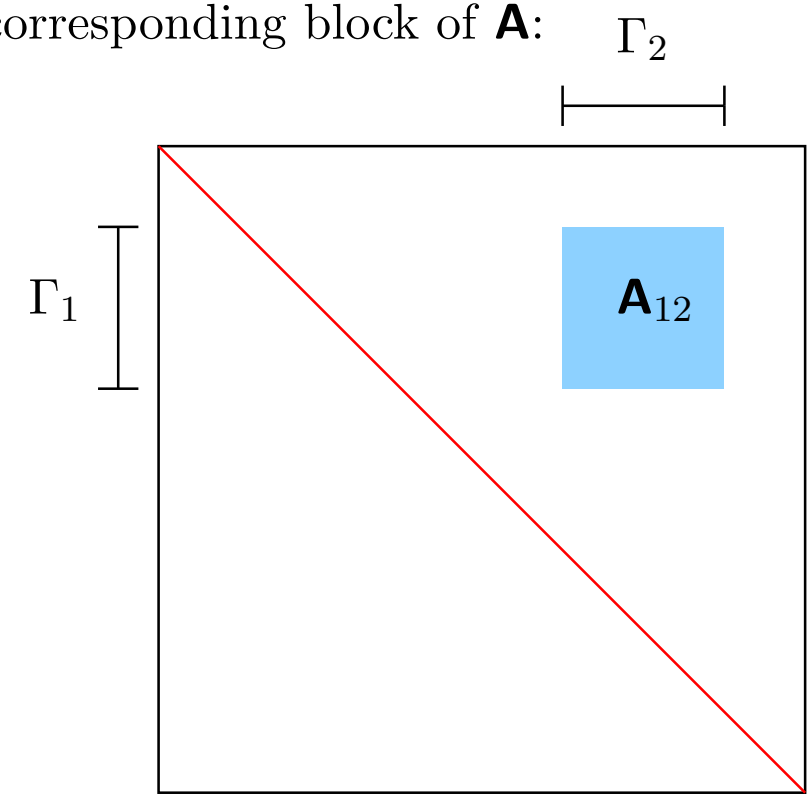
(without the diagonal entries)

Key observation: Off-diagonal blocks of **A** have low rank.

Consider two patches Γ_1 and Γ_2 and the corresponding block of **A**:



The contour Γ

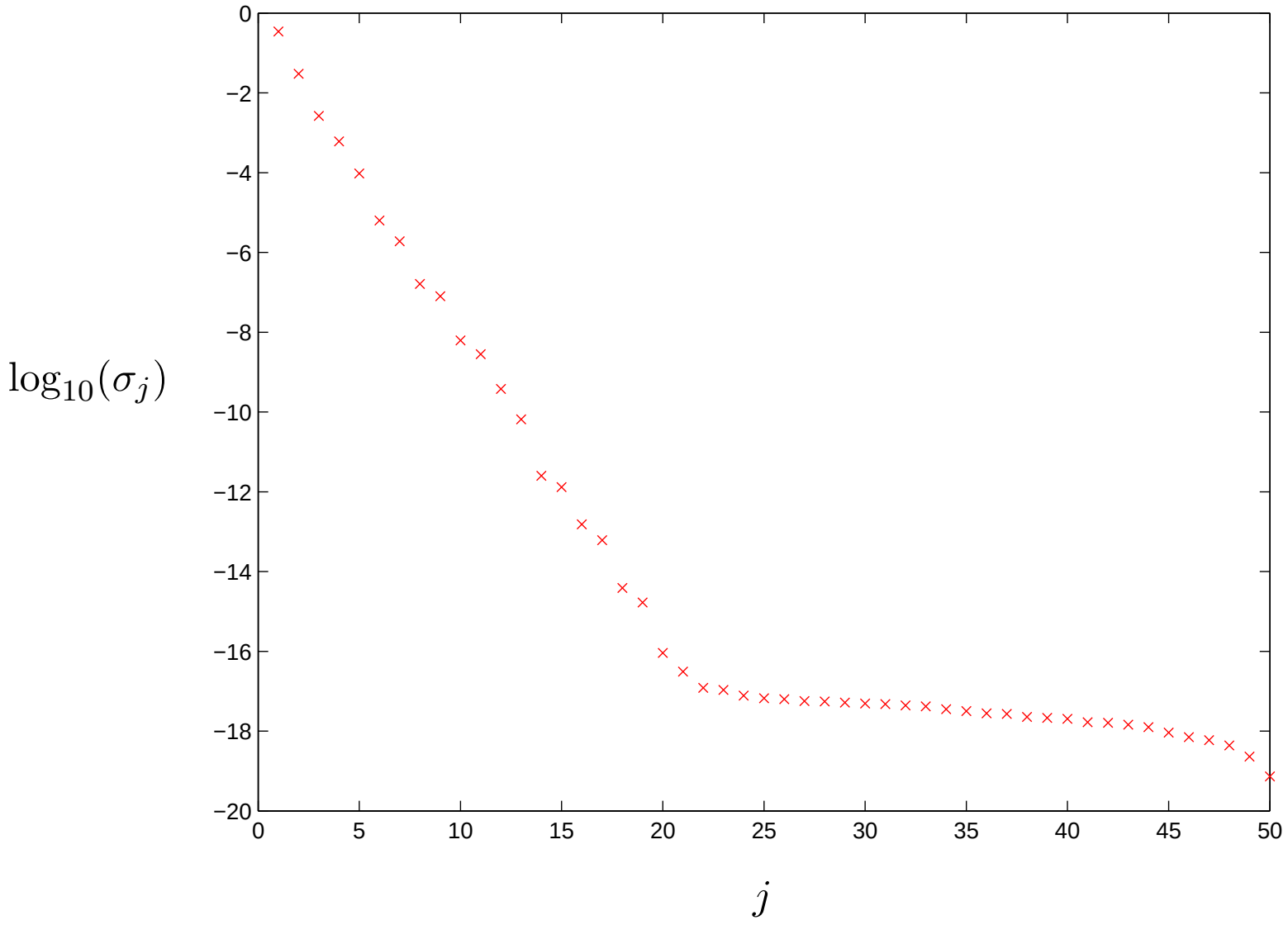


The matrix **A**

The block $\mathbf{A}_{12}$ is a discretization of the integral operator

$$[\mathcal{S}_{\Gamma_1 \leftarrow \Gamma_2}\, u](\mathbf{x}) = u(\mathbf{x}) + \int_{\Gamma_2} \log|\mathbf{x}-\mathbf{y}|\, u(\mathbf{y})\, ds(\mathbf{y}), \qquad \mathbf{x} \in \Gamma_1.$$

Singular values of $\mathbf{A}_{12}$ (now for a 200×200 matrix $\mathbf{A}$):



... should do an HSS block as well ...

What we see is an artifact of the *smoothing effect* of coercive elliptic differential equations; it can be interpreted as a *loss of information.*

This effect has many well known physical consequences:

- The intractability of solving the heat equation backwards.
- The *St Venant principle* in mechanics.
- The inaccuracy of imaging at sub-wavelength scales.

Such phenomena should be viewed in contrast to high-frequency scattering problems — extreme accuracy of optics *etc.*

Numerical examples:

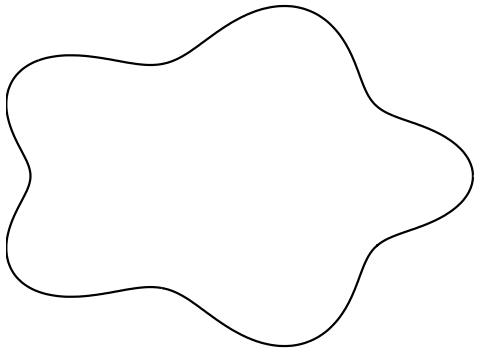
We invert a matrix approximating the operator

$$[A\,u](\mathbf{x}) = \frac{1}{2}\,u(\mathbf{x}) - \frac{1}{2\,\pi}\int_\Gamma D(\mathbf{x},\mathbf{y})\,u(\mathbf{y})\,ds(\mathbf{y}), \qquad \mathbf{x}\in\Gamma,$$

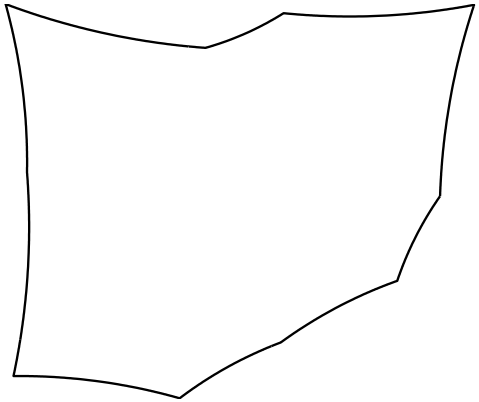
where D is the double layer kernel associated with Laplace's equation,

$$D(\mathbf{x},\mathbf{y}) = \frac{1}{2\pi}\,\frac{\mathbf{n}(\mathbf{y})\cdot(\mathbf{x}-\mathbf{y})}{|\mathbf{x}-\mathbf{y}|^2},$$

and where Γ is either one of the countours:



Smooth star



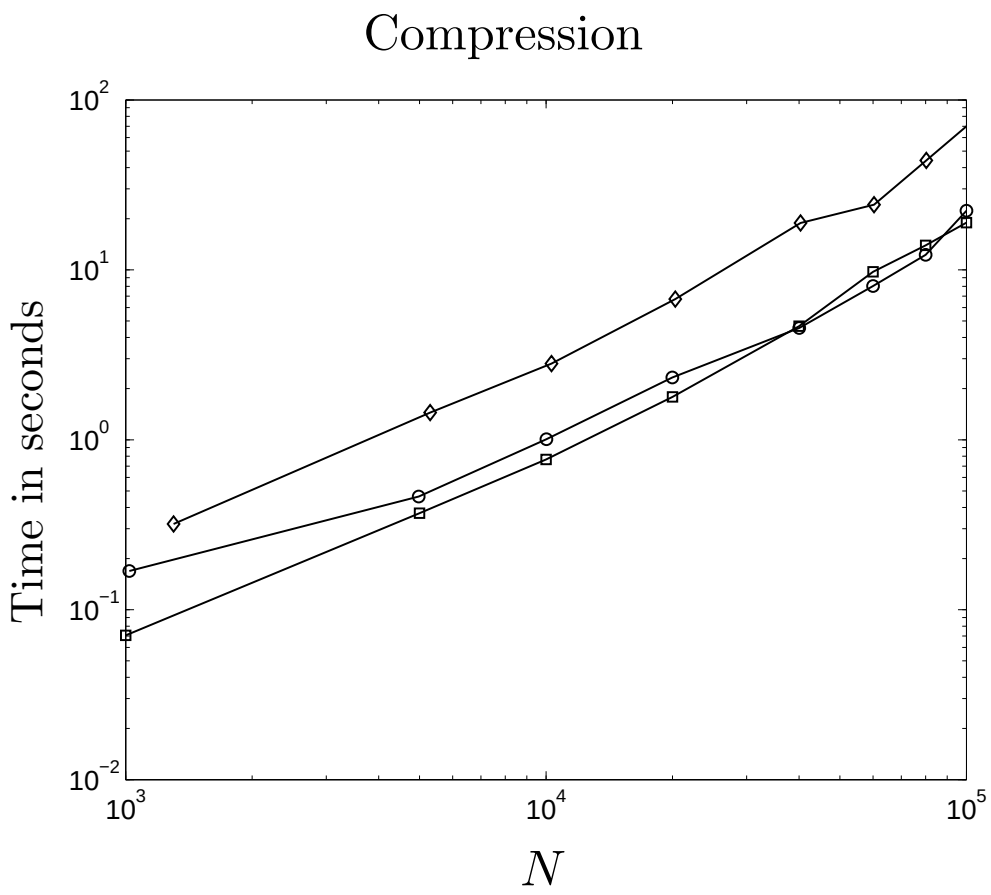
Star with corners

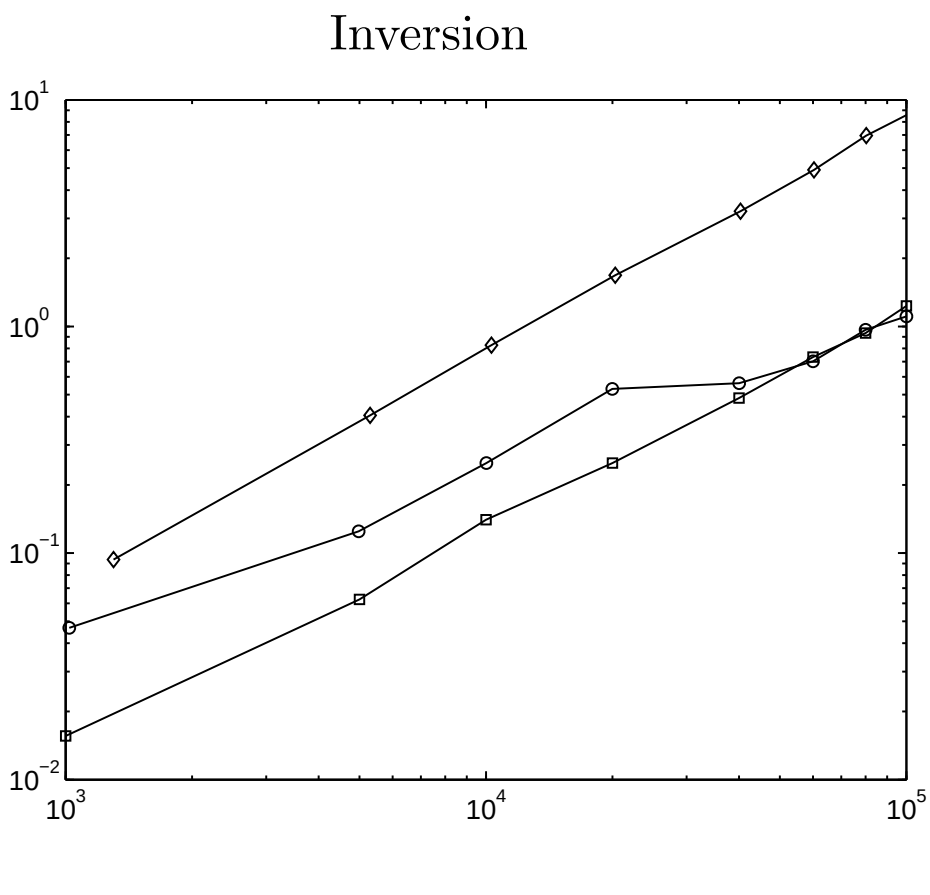
(local refinements at corners)

Snake

(# oscillations $\sim N$)

Examples from "A direct solver with O(N) complexity for integral equations on one-dimensional domains," A. Gillman, P. Young, P.G. Martinsson, 2011, in review.





The graphs give the times required for:

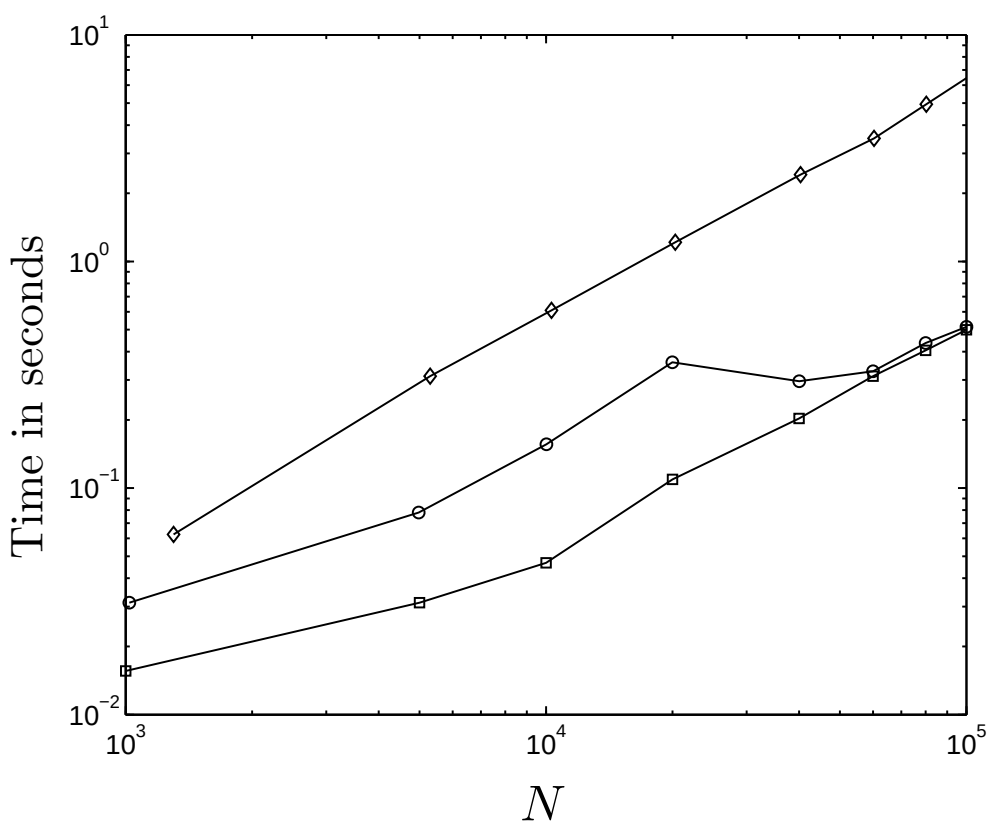
- Computing the HSS representation of the coefficient matrix.
- Inverting the HSS matrix.

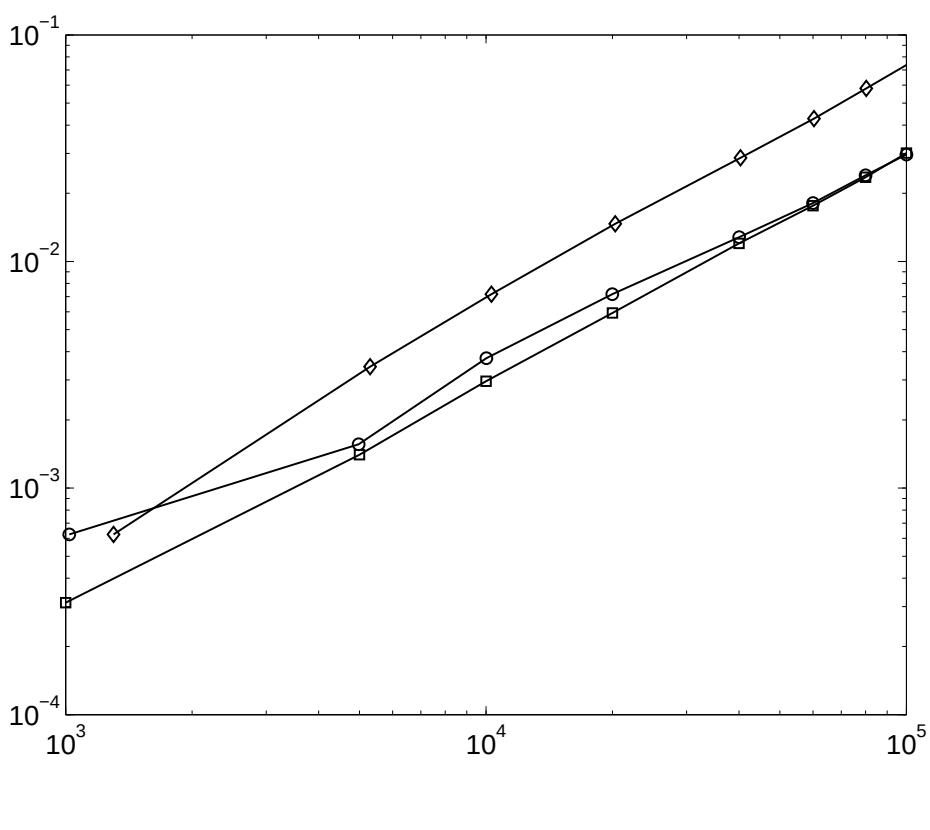
Within each graph, the three lines correspond to the three contours considered:

□ Smooth star.

○ Star with corners.

◇ Snake.





The graphs give the times required for:

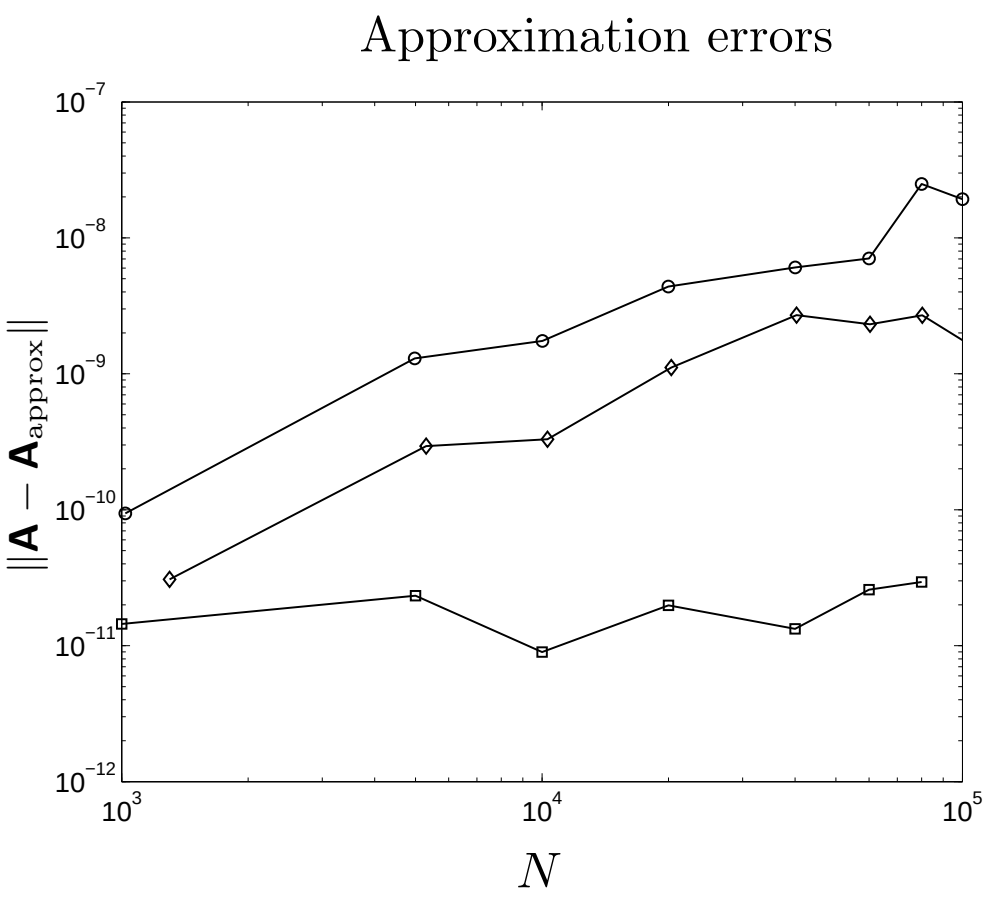
- Transforming the computed inverse to standard HSS format.
- Applying the inverse to a vector (i.e. solving a system).

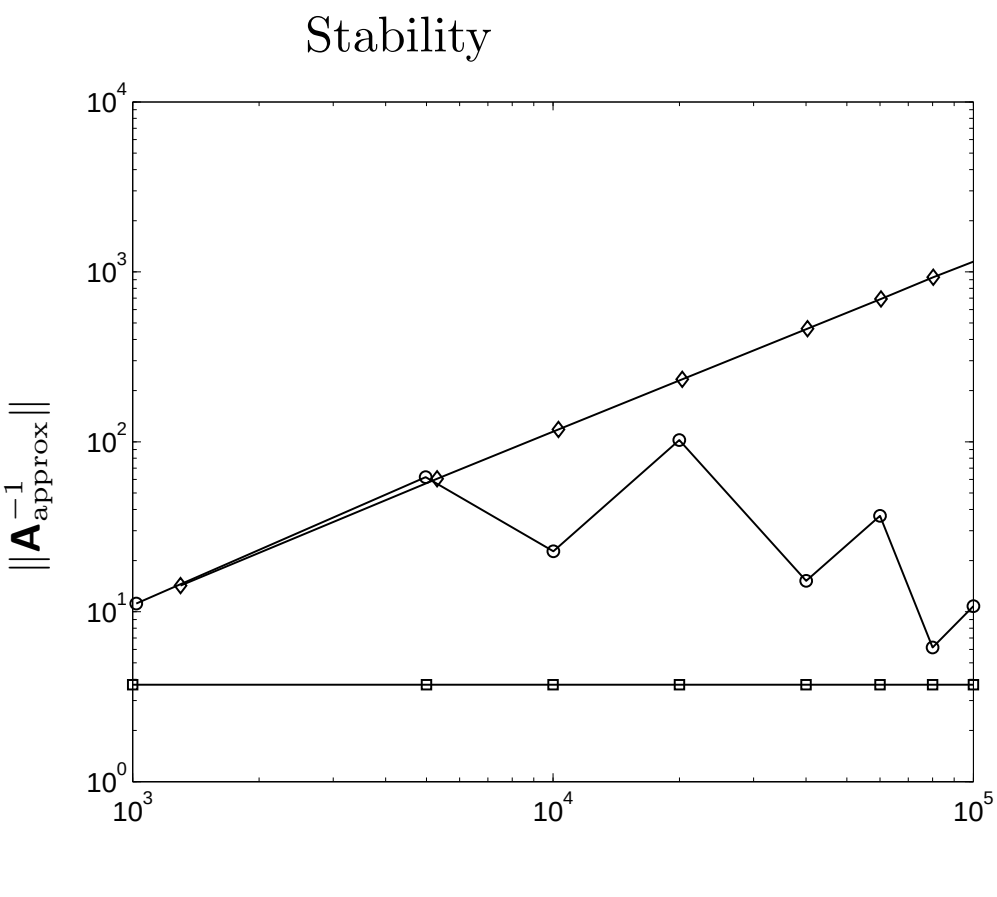
Within each graph, the three lines correspond to the three contours considered:

□ Smooth star.

○ Star with corners.

◇ Snake.



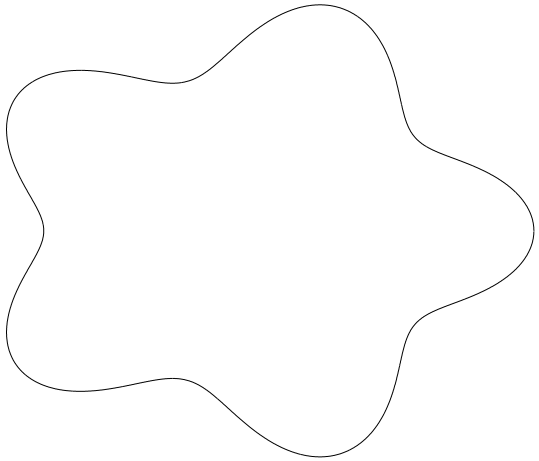


The graphs give the error in the approximation, and the norm of the inverse.

Within each graph, the three lines correspond to the three contours considered:

- □ Smooth star.
- ○ Star with corners.
- ◇ Snake.

Example: An interior Helmholtz Dirichlet problem

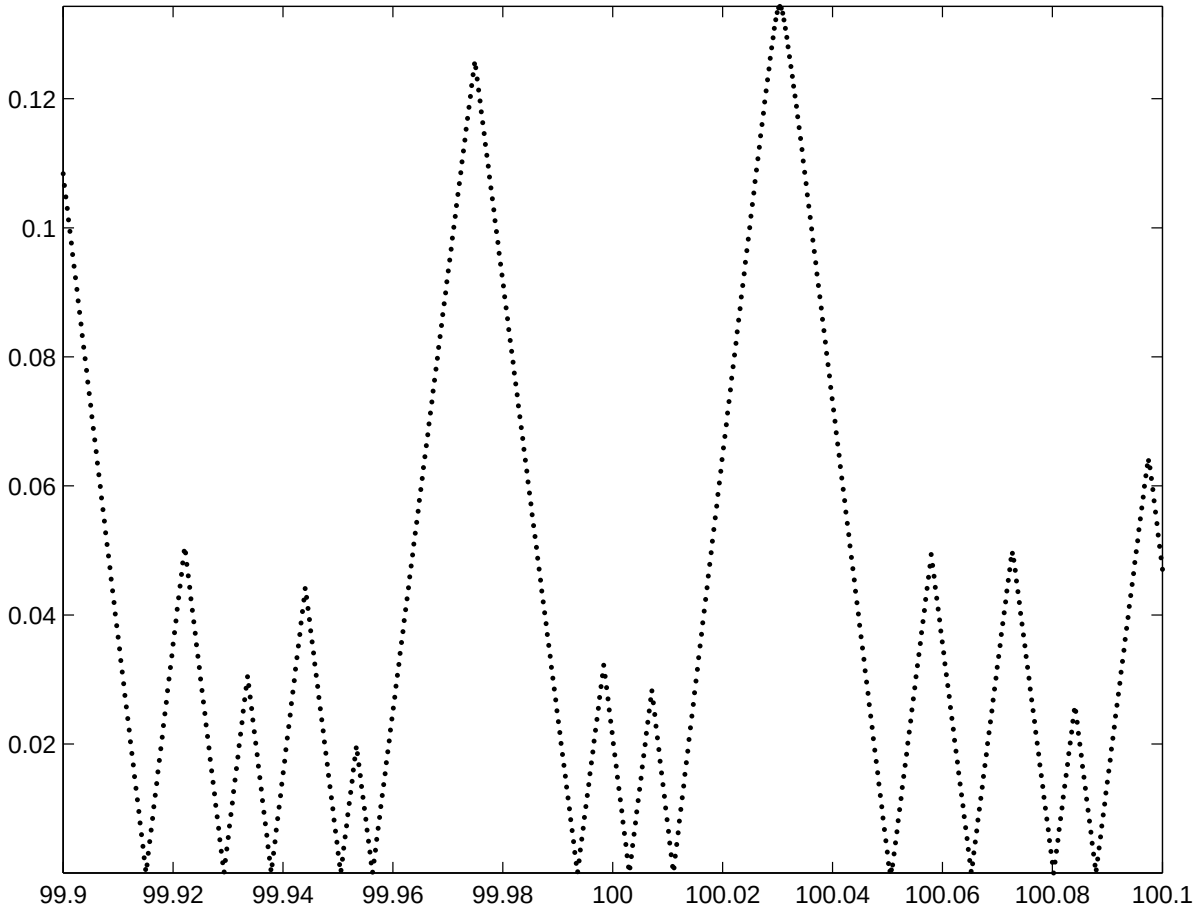


The diameter of the contour is about 2.5. An interior Helmholtz problem with Dirichlet boundary data was solved using $N = 6\,400$ discretization points, with a prescribed accuracy of 10^{-10}.

For $k = 100.011027569\cdots$, the smallest singular value of the boundary integral operator was $\sigma_{\min} = 0.00001366\cdots$.

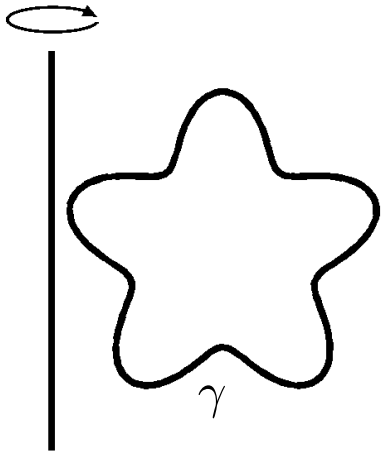
Time for constructing the inverse: 0.7 seconds.

Error in the inverse: 10^{-5}.

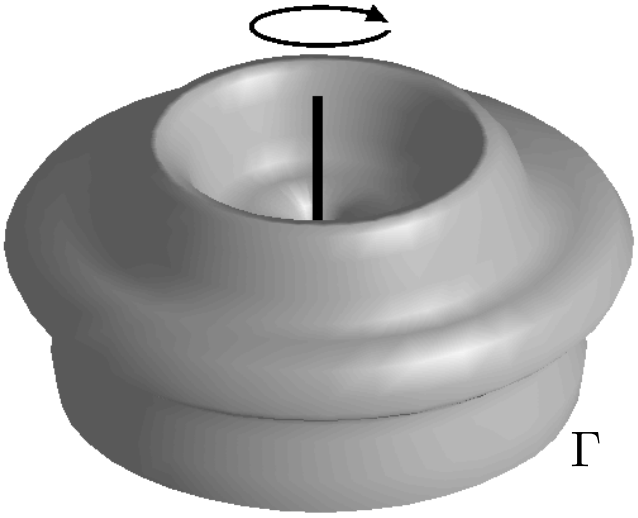


Plot of $\sigma_{\min}$ versus k for an interior Helmholtz problem on the smooth pentagram. The values shown were computed using a matrix of size $N = 6400$. Each point in the graph required about $60s$ of CPU time.

Example: BIEs on rotationally symmetric surfaces (with Patrick Young)



Generating curve



Surface

Let Γ be a surface of rotation generated by a curve γ, and consider a BIE associated with Laplace's equation:

$$(3) \qquad \frac{1}{2}\sigma(\mathbf{x}) + \int_\Gamma \frac{\mathbf{n}(\mathbf{y})\cdot(\mathbf{x}-\mathbf{y})}{4\pi|\mathbf{x}-\mathbf{y}|^3}\,\sigma(\mathbf{y})\,dA(\mathbf{y}) = f(\mathbf{x}). \quad \mathbf{x}\in\Gamma$$

To (3), we apply the Fourier transform in the azimuthal angle (executed computationally via the FFT) and get

$$\frac{1}{2}\sigma_n(\mathbf{x}) + \int_\gamma k_n(\mathbf{x},\mathbf{y})\,\sigma_n(\mathbf{y})\,dl(\mathbf{y}) = f_n(\mathbf{x}), \quad \mathbf{x}\in\gamma, \qquad n\in\mathbb{Z}.$$

Then discretize the sequence of equations on γ using the direct solvers described (with special quadratures, *etc*).

We discretized the surface using 400 Fourier modes, and 800 points on γ for a total problem size of

$$N = 320\,000.$$

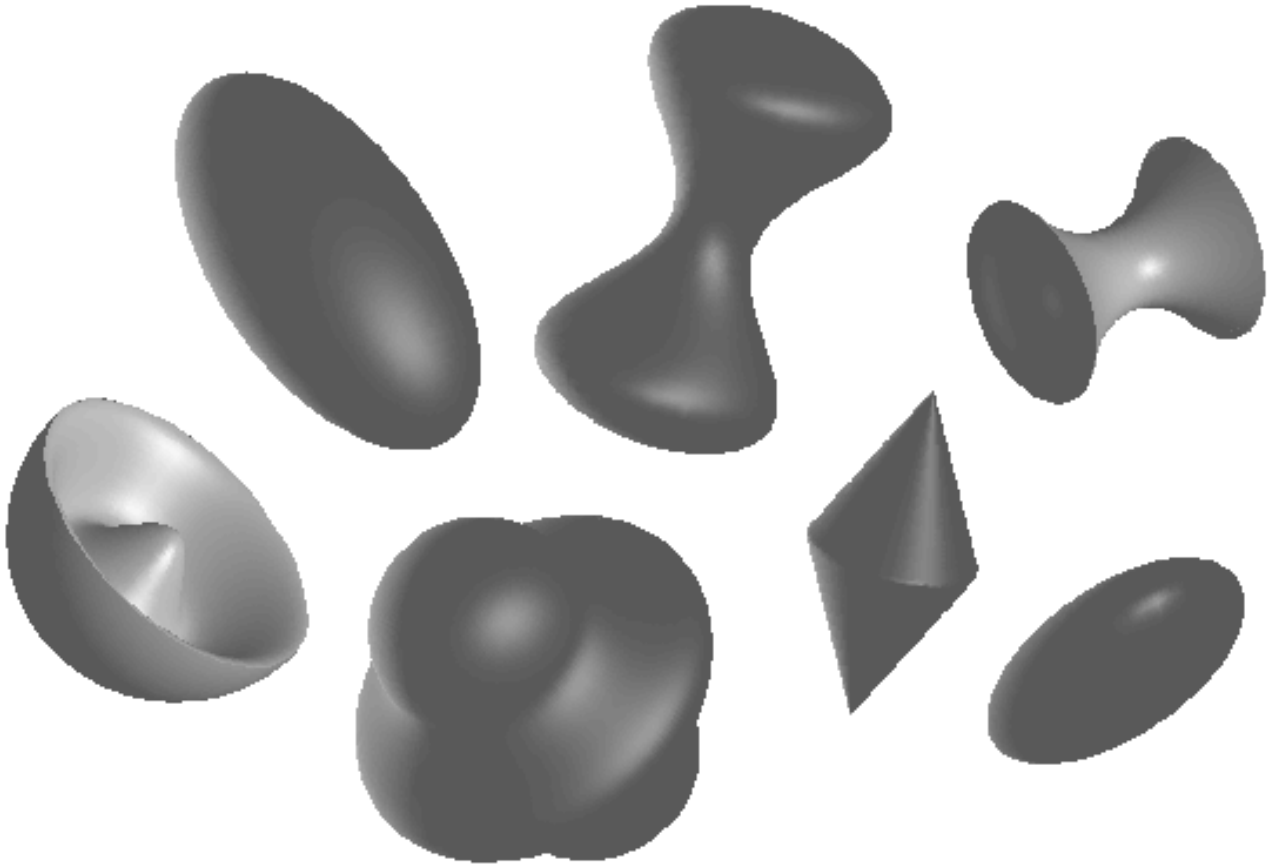
For typical loads, the relative error was less than 10^{-10} and the CPU times were

$$T_{\text{invert}} = 2\text{min} \qquad T_{\text{solve}} = 0.3\text{sec}.$$

Work in progress (with Sijia Hao): Extension to multibody acoustic scattering:

Individual scattering matrices are constructed via a relatively expensive pre-computation.

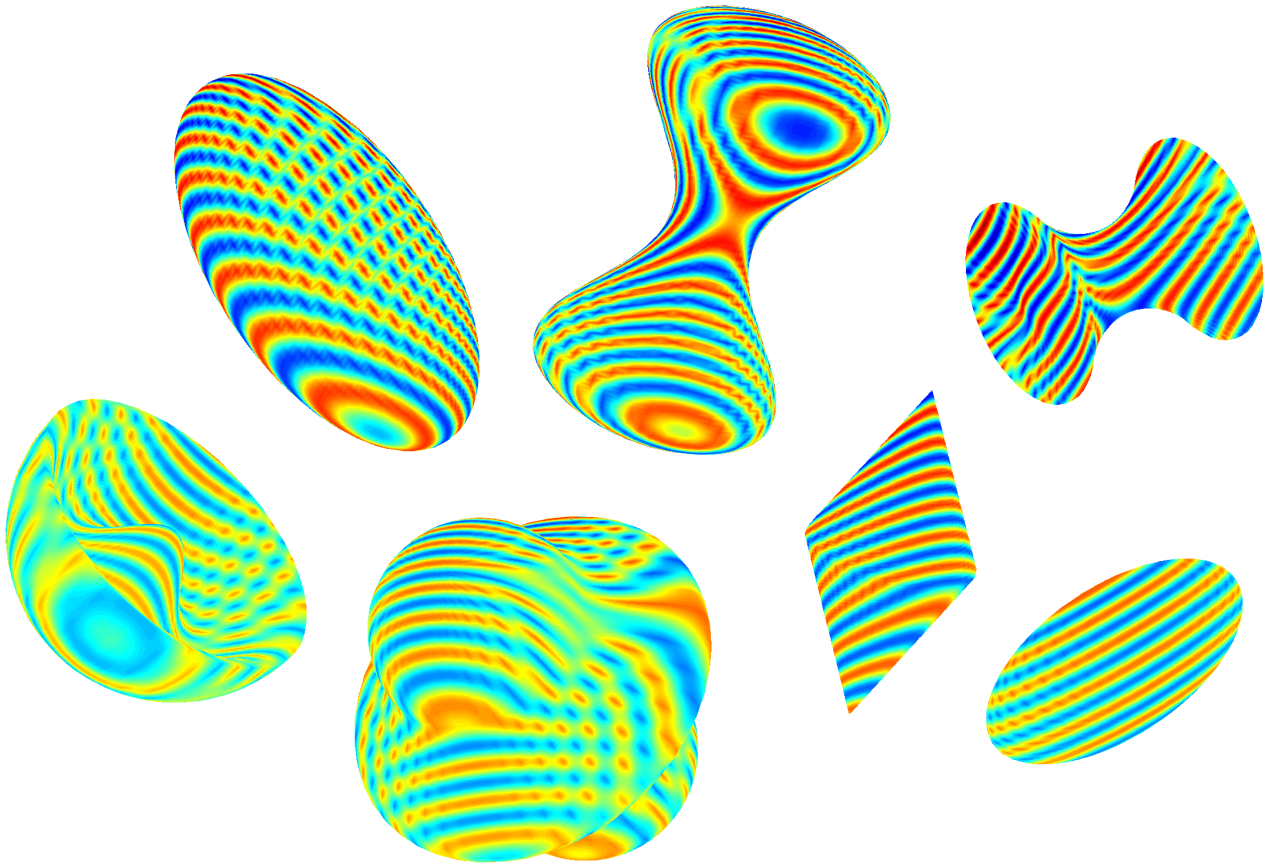
Inter-body interactions are handled via the wideband FMM and an iterative solver.



Work in progress (with Sijia Hao): Extension to multibody acoustic scattering:

Individual scattering matrices are constructed via a relatively expensive pre-computation.

Inter-body interactions are handled via the wideband FMM and an iterative solver.



Sample environment: Volume problems in 2D

Example: Consider an elliptic boundary value problem with variable coefficients

$$\text{(BVP)} \qquad \begin{cases} -\nabla \cdot (B(\mathbf{x})\, \nabla u(\mathbf{x})) + c(\mathbf{x})\, u(\mathbf{x}) = 0, & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases}$$

where Ω is a domain in $\mathbb{R}^2$ with piecewise smooth boundary Γ.
(Assume the operator is coercive, B and c need not be smooth.)

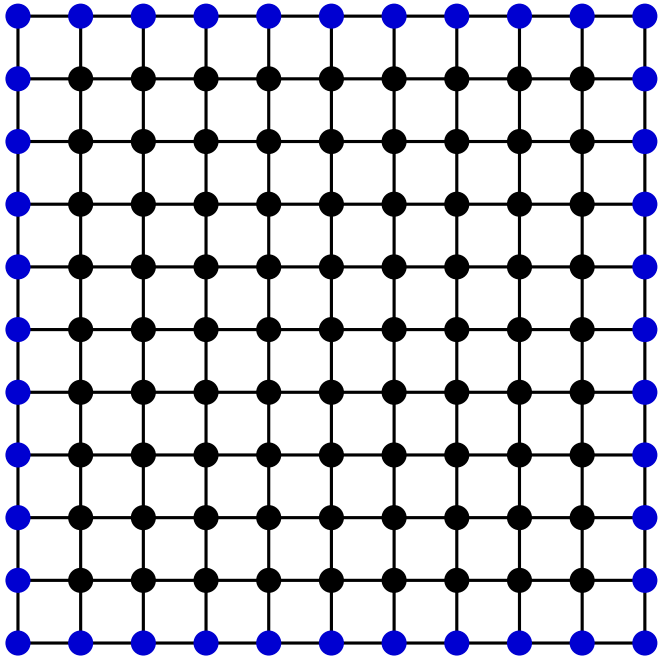
Let $\mathbf{A}$ denote the $N \times N$ matrix arising from an FD or FEM discretization of (BVP).
While $\mathbf{A}$ is sparse, its inverse (and LU factors) are dense.
However, they are highly compressible, and can be computed in $O(N)$ operations.

Note: The inverse of $\mathbf{A}$ mimics the action of the solution operator

$$u(\mathbf{x}) = \int_\Gamma G(\mathbf{x}, \mathbf{y})\, f(\mathbf{y})\, dA(\mathbf{y}),$$

where G is the Green's function of the problem.
(Note that G is known analytically only for the most trivial domains Ω.)

Example: **Inversion of a "Finite Element Matrix" (with A. Gillman)**



A grid conduction problem — $\mathbf{A}$ is a "five-point stencil" — very large, sparse.

Each link has conductivity drawn from a uniform random distribution on [1, 2].

Solution strategy: Perform nested dissection on the grid. Use HSS algebra to accelerate all computations involving dense matrices larger than a certain threshold. Total complexity is $O(N)$ (as compared to $O(N^{1.5})$ for classical nested dissection).

N	T_{solve} (sec)	T_{apply} (sec)	M (MB)	e_3	e_4
512^2	7.98	0.007	8.4	$2.7523e-6$	$6.6631e-9$
1024^2	26.49	0.014	18.6	-	-
2048^2	98.46	0.020	33.1	-	-
4096^2	435.8	0.039	65.6	-	-

T_{solve} Time required to compute all Schur complements ("set-up time")

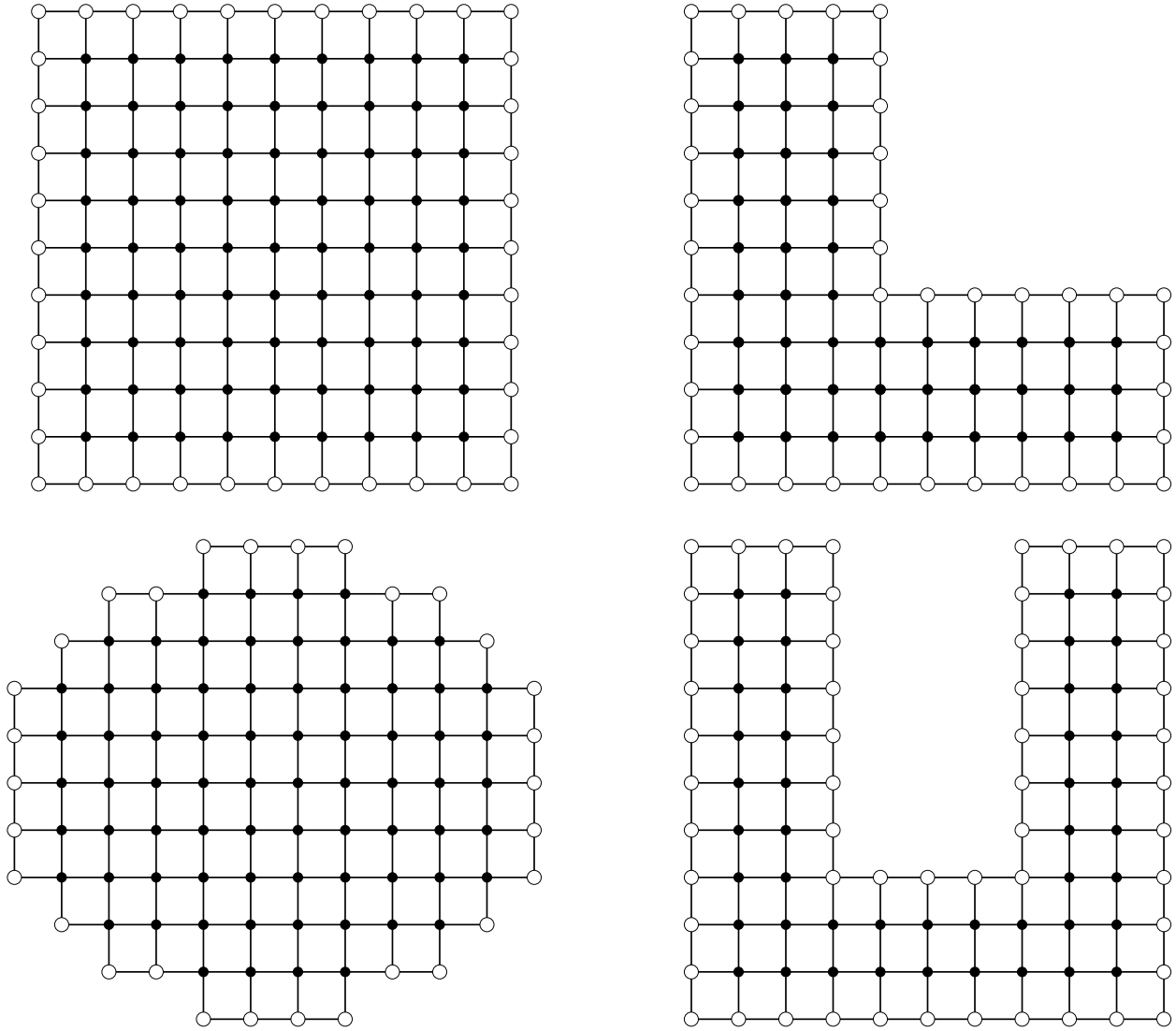
T_{apply} Time required to apply a Dirichlet-to-Neumann op. (of size $4\sqrt{N} \times 4\sqrt{N}$)

M Memory required to store the solution operator

e_3 The l^2-error in the vector $\tilde{\mathbf{A}}_{nn}^{-1} r$ where r is a unit vector of random direction.

e_4 The l^2-error in the first column of $\tilde{\mathbf{A}}_{nn}^{-1}$.

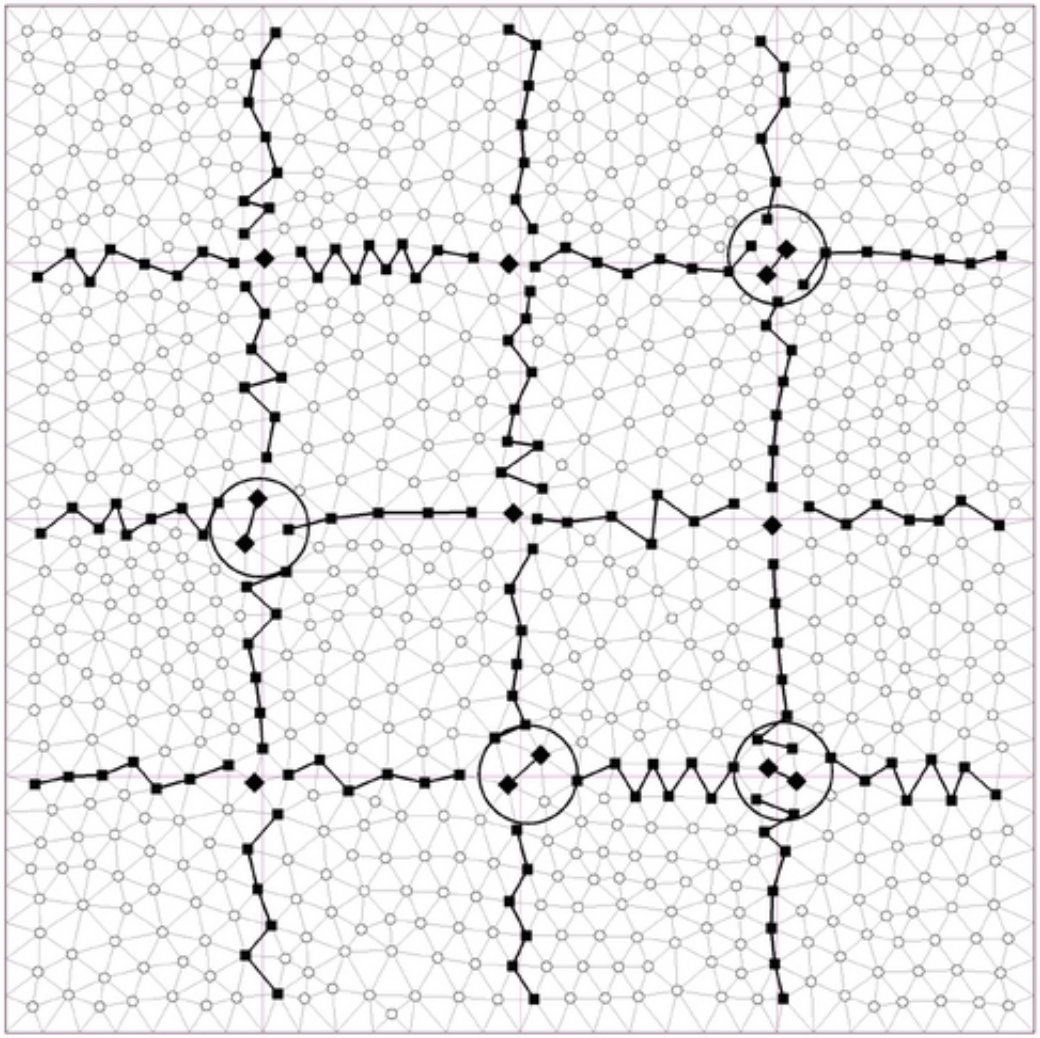
Note: Similar work by S. Chandrasekharan, M. Gu, X.S. Li, J. Xia; L. Grasedyck, R. Kriemann, S. LeBorne; P. Schmitz and L. Ying; E. Michielssen; ...



Fun fact: For *constant coefficient* difference operators on regular grids, the Dirichlet-to-Neumann operator for a general domain can be computed in $O(\sqrt{N})$ operations.
$N = 10^{12}$ can be handled with ease on a laptop.

See Gillman & Martinsson, JCP, 229(24), pp. 9026–9041, 2010.

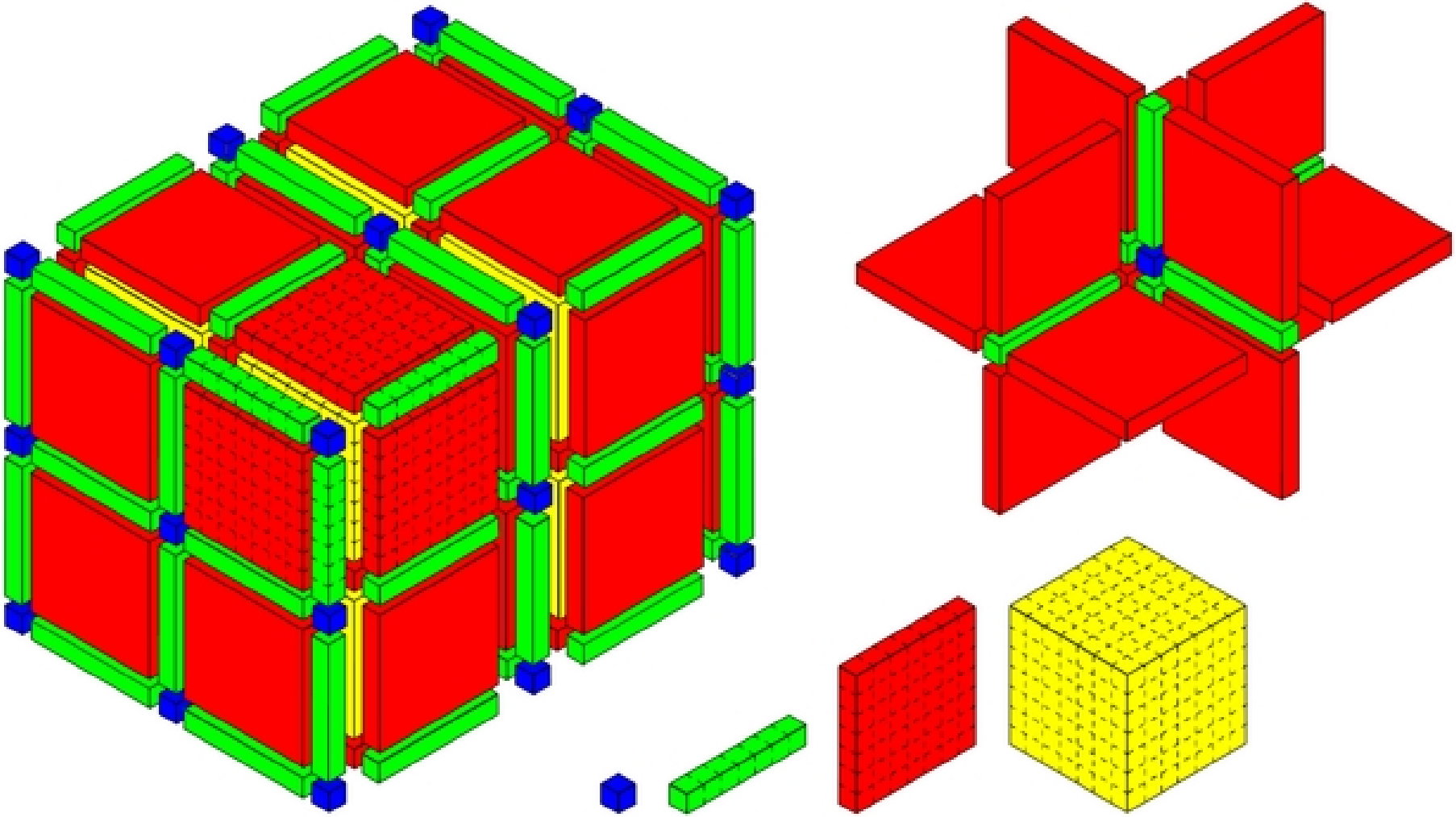
Example: $O(N)$ nested dissection on general meshes in $\mathbb{R}^2$:



From *A fast direct solver for elliptic problems on general meshes in 2D* by P. Schmitz and L. Ying, 2010.

Related work by S. Chandrasekharan, M. Gu, X.S. Li, J. Xia; Grasedyck & LeBorne; etc.

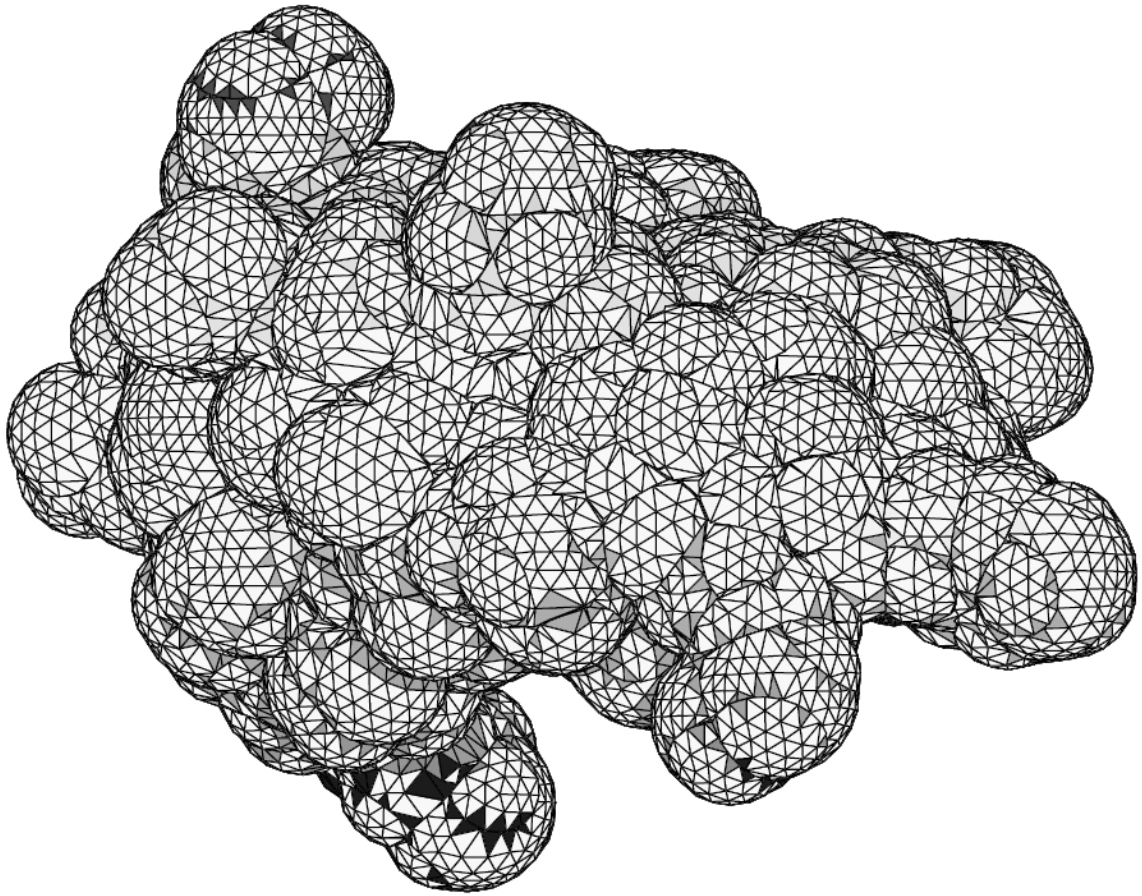
Example: Accelerated nested dissection on grids in $\mathbb{R}^3$:



From *A fast direct solver for elliptic problems on Cartesian meshes in 3D* by P. Schmitz and L. Ying, 2010.

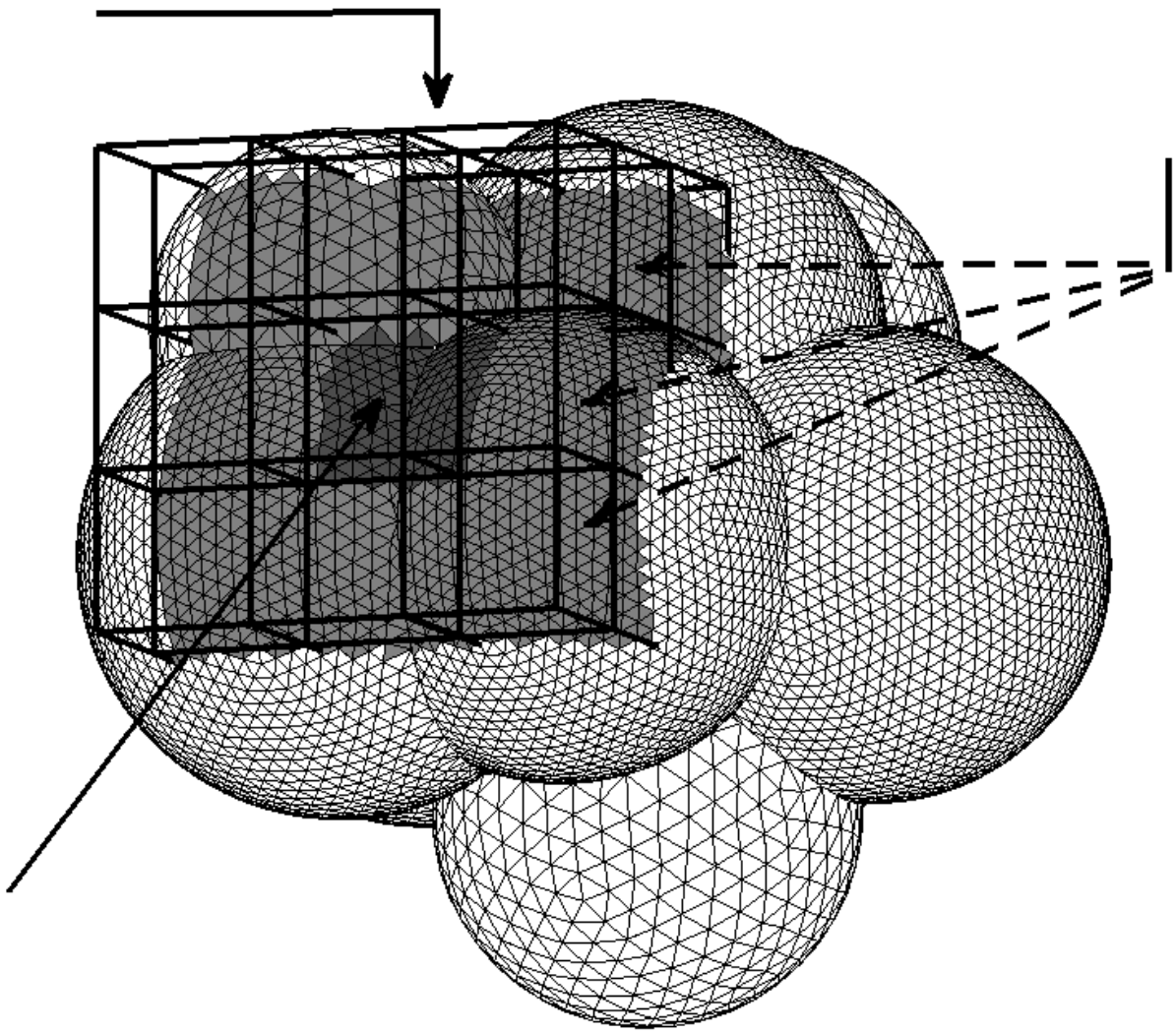
Related work by J. Xia et al; L. Grasedyck & S. LeBorne; E. Michielssen; etc.

Sample environment: BIEs on surfaces in $\mathbb{R}^3$



The scheme for BIEs in $\mathbb{R}^2$ can without modifications be applied to BIEs in $\mathbb{R}^3$.
However, the ranks then grow for larger patches, and the asymptotic complexity will be:

Inversion step:	$O(N^{1.5})$ (with small scaling constant)
Application of the inverse:	$O(N)$

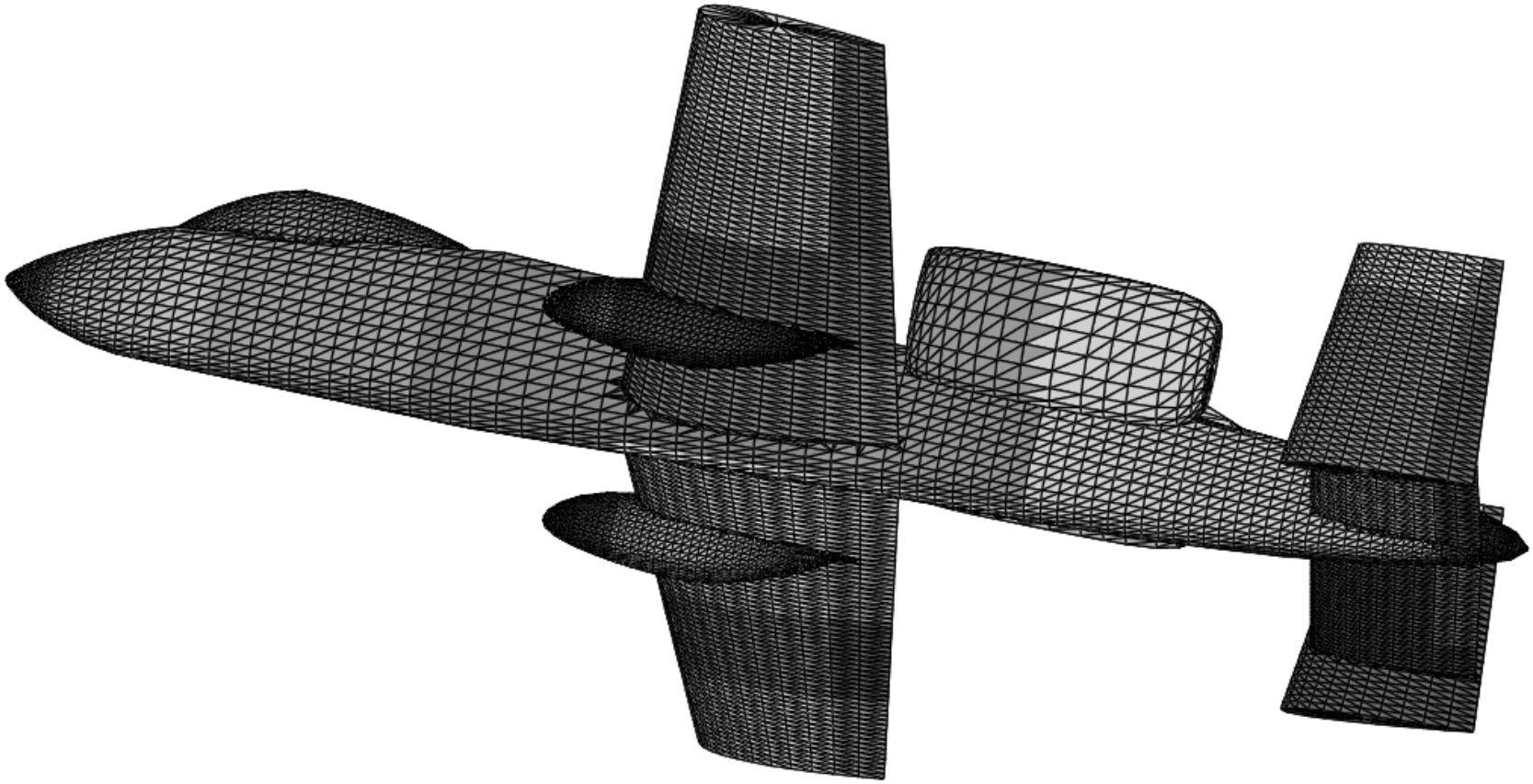


All geometric operations are now done with respect to an oct-tree.

It is sparsely populated since the object being discretized is two-dimensional.

Example: Triangulated aircraft

Computation carried out by Denis Gueyffier at Courant.

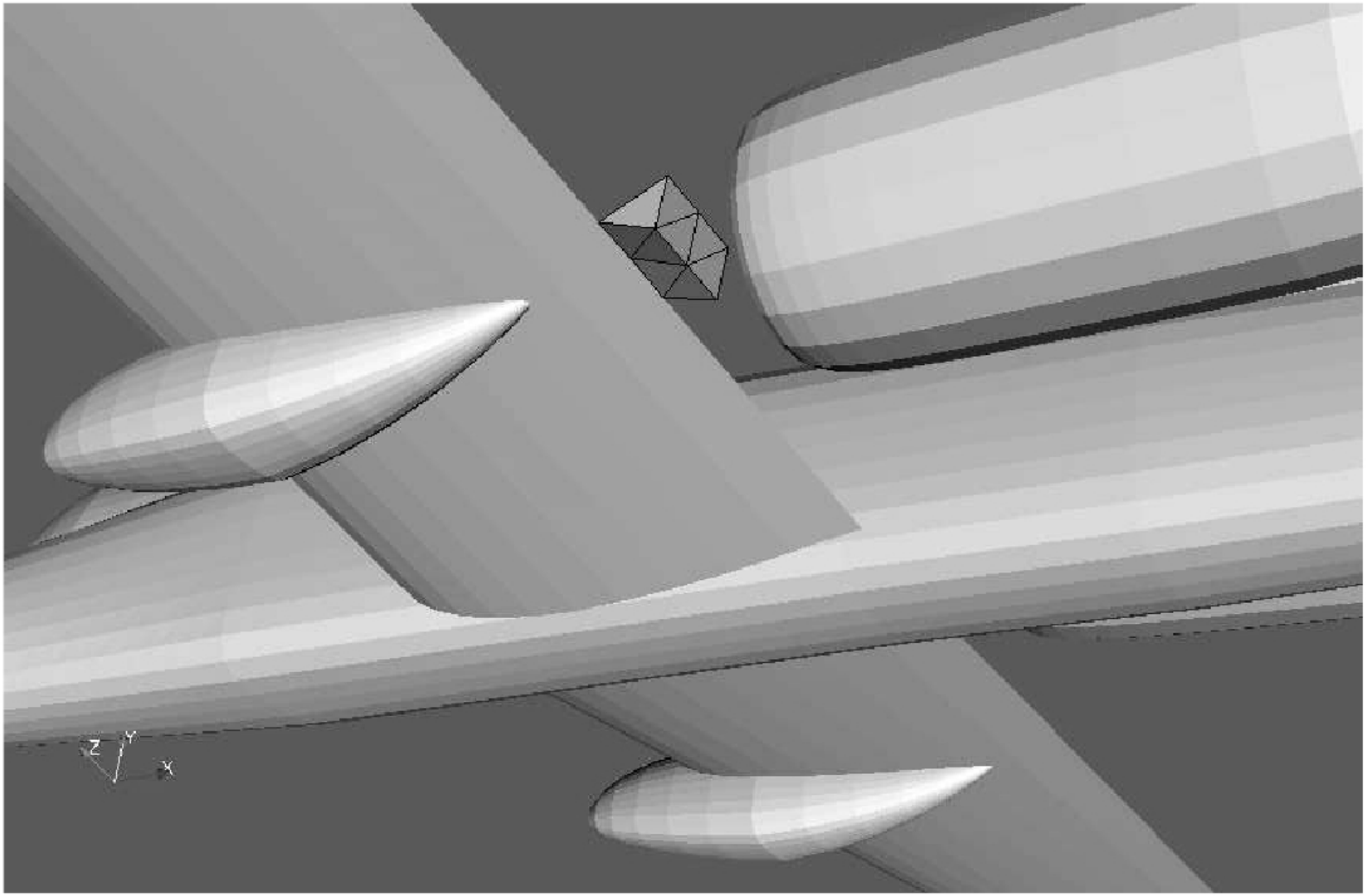


Laplace's equation. 28 000 triangles. Standard office desktop.

Cost of very primitive inversion scheme (low accuracy, etc.): 15 min

Cost of applying the inverse: 0.2 sec

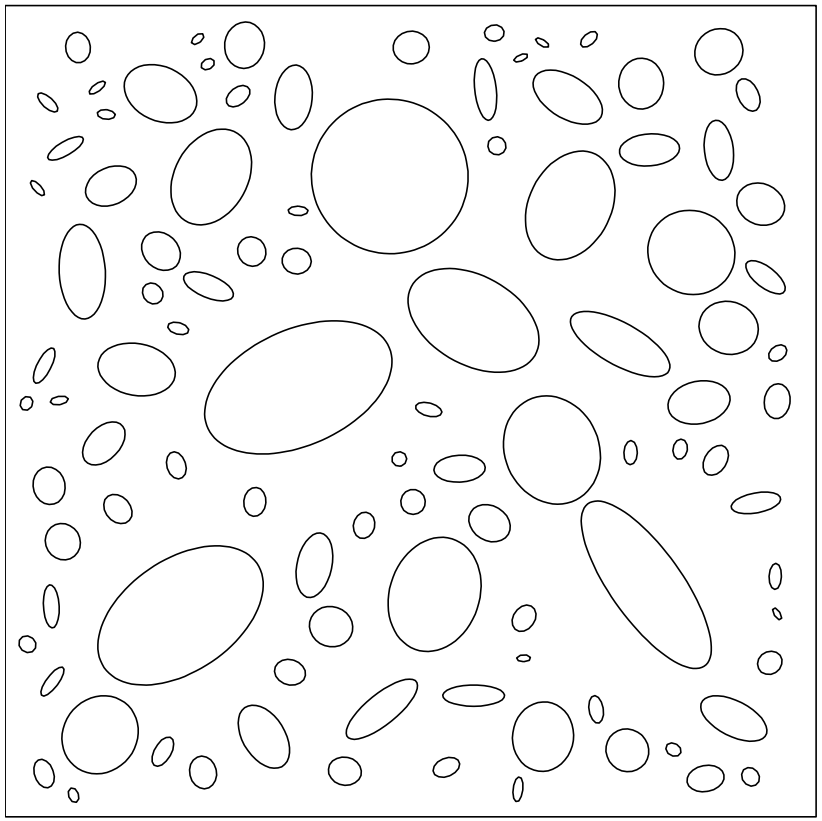
From *Fast direct solvers for integral equations in complex three-dimensional domains*, by Greengard, Gueyffier, Martinsson, Rokhlin, Acta Numerica 2009.



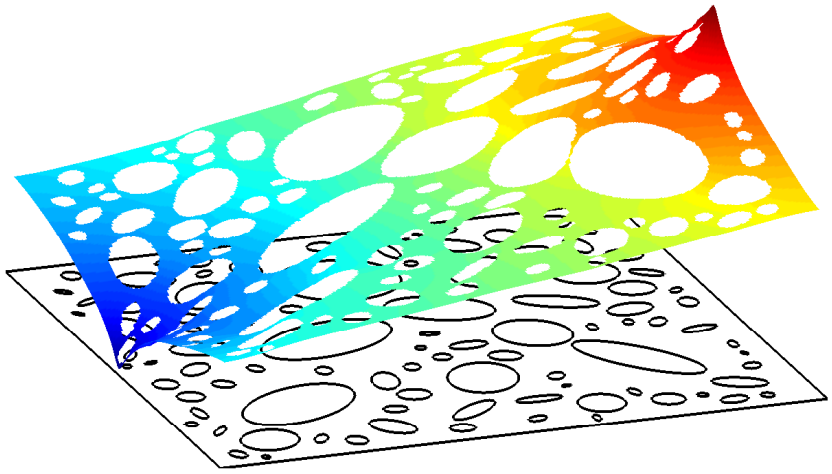
Observation: Local updates to the geometry are very cheap. Adding a (not so very aerodynamic) flap corresponds to a rank-15 update and can be done in a fraction of a second.

Note: While our codes are very primitive at this point, there exist extensive $\mathcal{H}/\mathcal{H}^2$-matrix based libraries with better asymptotic estimates for inversion. `www.hlib.org`

A conduction problem on a perforated domain



Geometry



Potential

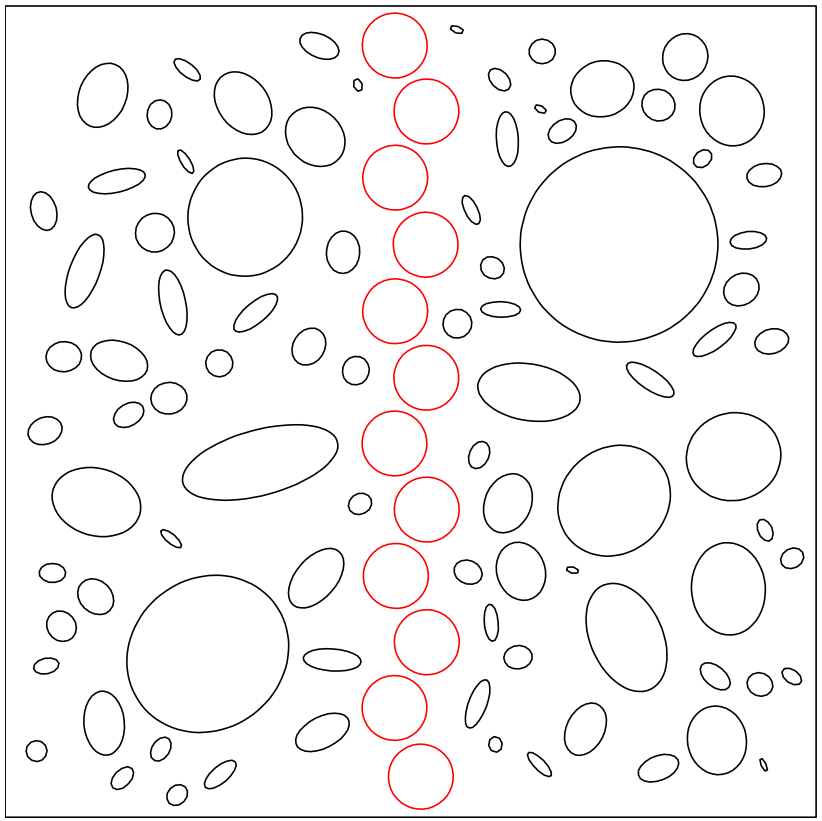
The Neumann-to-Dirichlet operator for the exterior boundary was computed.

The boundary was split into 44 panels, with 26 Gaussian quadrature nodes on each one.

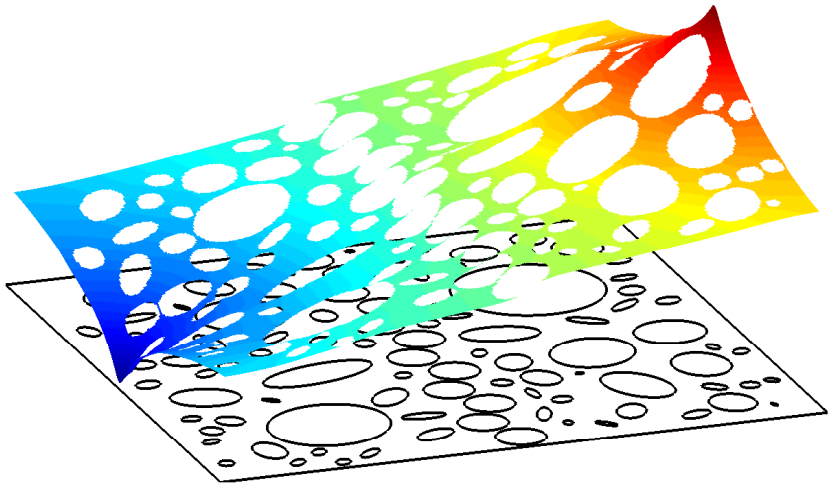
This gives a relative accuracy of 10^{-10} for evaluating fields at points very close to the boundary (up to 0.5% of the side-length removed).

Storing the N2D operator (in a data-sparse format) requires 120 floats per degree of freedom.

A conduction problem on a perforated domain — close to "percolation"



Geometry



Potential

The Neumann-to-Dirichlet operator for the exterior boundary was computed.

The boundary was split into 44 panels, with 26 Gaussian quadrature nodes on each one.

This gives a relative accuracy of 10^{-10} for evaluating fields at points very close to the boundary (up to 0.5% of the side-length removed).

Storing the N2D operator (in a data-sparse format) requires 118 floats per degree of freedom.

Observation:

Dense matrices that arise in numerical algorithms for elliptic PDEs are surprisingly well suited to the HSS-representation. The format is robust to:

- Irregular grids.
- PDEs with non-smooth variable coefficients.
- Inversion, LU-factorization, matrix-matrix-multiplies, etc.

For oscillatory problems, the ranks grow as the wave-length of the problem is shrunk relative to the size of the geometry, which eventually renders the direct solvers prohibitively expensive. However, the methodology remains efficient for "surprisingly" small wave-lengths.

Some supporting theory and "intuitive arguments" exist, but the observed performance still exceeds what one would expect, both in terms of the range of applicability and what the actual ranks should be. (At least what I would expect!)

Additional talks on this subject:

- Dan Jiao: MS32, Monday 6:00pm - 6:25pm.
- Adrianna Gillman: CP20, Thursday 4:10pm - 4:30pm.

Assertions:

- Fast direct solvers excel for problems on 1D domains. (They should become the default.)
  - Integral operators on the line.
  - Boundary Integral Equations in $\mathbb{R}^2$.
  - Boundary Integral Equations on rotationally symmetric surfaces in $\mathbb{R}^3$.
- Existing fast direct solvers for "finite element matrices" associated with elliptic PDEs in $\mathbb{R}^2$ work very well. In $\mathbb{R}^3$, they can be game-changing in specialized environments.

Predictions:

- For BIEs associated with non-oscillatory problems on surfaces in $\mathbb{R}^3$, the complexity will be reduced from $O(N(\log N)^p)$ to $O(N)$, with a modest scaling constant.
- *Randomized methods* will prove enormously helpful.
  They have already demonstrated their worth in large scale linear algebra.
- Direct solvers for *scattering problems* will find users, even if expensive.
  $O(N^{1.5})$ or $O(N^2)$ flop counts may be OK, provided parallelization is possible.
- Direct solvers will provide a fantastic tool for *numerical homogenization.*

Open questions:

- How efficient can direct solvers be for volume problems in 3D?
- Are $O(N)$ direct solvers for highly oscillatory problems possible?