Fast and accurate techniques for computing Schur complements and performing numerical coarse graining

Gunnar Martinsson
The University of Colorado at Boulder

Students:

Adrianna Gillman

Nathan Halko

Patrick Young

Collaborators:

Vladimir Rokhlin (Yale)

Mark Tygert (Courant)

Model problem: Let Ω be a domain in $\mathbb{R}^d$ with boundary Γ. For $y \in \Gamma$, let $n(y)$ denote a surface normal. Let $a = a(x)$ be a function on Ω, and consider the BVP:

$$
\left\{
\begin{aligned}
Au(x) =& \quad 0, && x \in \Omega,\\
\frac{\partial u(x)}{\partial n} =& \quad h(x), && x \in \Gamma,
\end{aligned}
\right.
\tag{1}
$$

where

$$Au(x) = -\nabla \cdot \big(a(x)\nabla u(x)\big).$$

We suppose that $a(x)$ varies on a scale smaller than the scale of Ω.

Coarse graining: Find an operator A_{coarse} such that the solution to

$$
\left\{
\begin{aligned}
A_{\text{coarse}}v(x) =& \quad 0, && x \in \Omega,\\
\frac{\partial v(x)}{\partial n} =& \quad h(x), && x \in \Gamma,
\end{aligned}
\right.
\tag{2}
$$

is in some sense "close" to the solution of (1) for some class of functions h, and such that (2) is easier to solve than (1).

Model problem: Let Ω be a domain in $\mathbb{R}^d$ with boundary Γ. For $y \in \Gamma$, let $n(y)$ denote a surface normal. Let $a = a(x)$ be a function on Ω, and consider the BVP:

$$(3) \qquad \begin{cases} -\nabla \cdot \big(a(x)\nabla u(x)\big) = 0, & x \in \Omega, \\ \dfrac{\partial u(x)}{\partial n} = h(x), & x \in \Gamma. \end{cases}$$

Analytic solution: There exists a function H such that

$$u(x) = \int_\Gamma H(x,y)\,h(y)\,ds(y) = [\mathbf{H}\,h](x), \qquad x \in \Gamma.$$

The operator $\mathbf{H}$ is known as the *Neumann-to-Dirichlet operator.*

Coarse graining: Find $\mathbf{H}_{\text{coarse}}$ such that

$$\mathbf{H}\,h \approx \mathbf{H}_{\text{coarse}}\,h$$

for some class of functions h.

Model problem: $$\left\{\begin{aligned} -\nabla\cdot\big(a(x)\nabla u(x)\big) &= 0, \qquad && x\in\Omega,\\ \frac{\partial u(x)}{\partial n} &= h(x), && x\in\Gamma. \end{aligned}\right.$$

Analytic solution: $$u(x) = \int_\Gamma H(x,y)\,h(y)\,ds(y) = [\mathbf{H}\,h](x), \qquad x\in\Gamma.$$

Claims:

- The operator $\mathbf{H}$ has a *data-sparse* approximate representation $\mathbf{H}_{\text{approx}}$:
  - Numerically sparse in a wavelet basis.
  - It is an $\mathcal{H}$-matrix. (Even $\mathcal{H}^2$, typically.)
  - It can be applied rapidly using the Fast Multipole Method (FMM).
- The data-sparse approximation can be computed. This currently requires the micro-structure to be fully resolved, but efficient numerical methods exist.

Speculation:

- It seems possible to compute $\mathbf{H}_{\text{approx}}$ without resolving the micro-structure.

Why construct an approximation to the Neumann-to-Dirichlet operator:

- It is an excellent "reduced model" for the geometry:
  - Every scale interacts on its appropriate level.
  - Highly accurate even near boundaries and concentrated loads.
  - Efficient representation of phenomena such as "percolating micro-structures."
- Having the solution operator is very practical — a "solve" is just an operator application.
- The N-to-D operator can be computed and stored once for a given geometry. It is cheap to update so excellent for engineering design.
- Operator algebra for N-to-D operators: They can be merged, added, introduced to handle one part of a multi-physics simulation, *etc.*

In describing the methodology, we will actually address the broader problem of how to compute an approximation to the solution operator of

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

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

- Linear elasticity.
- Stokes' equation.
- Helmholtz' equation (low frequencies).
- The Yukawa equation.

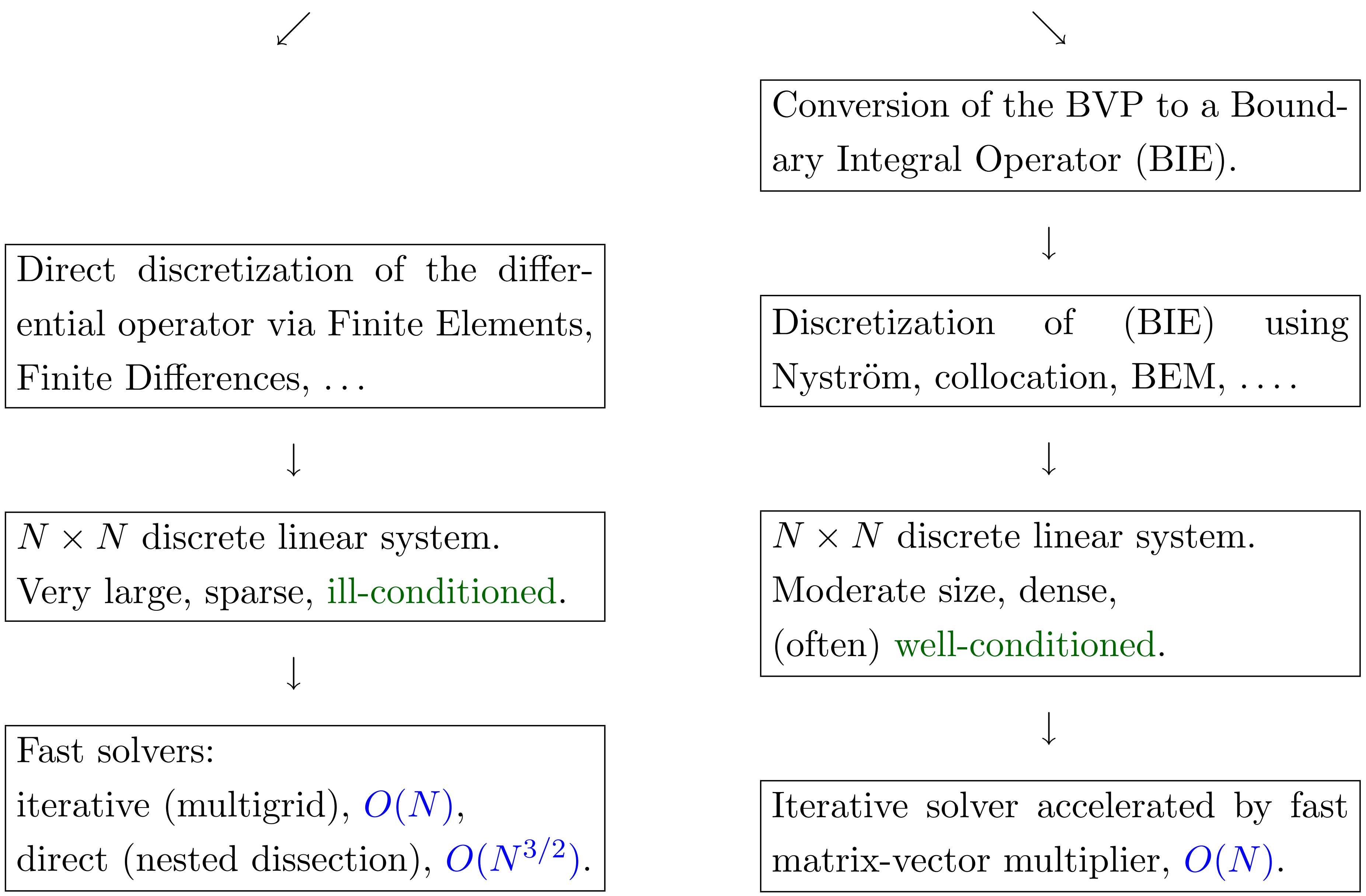
The full solution operator has the analytic form

$$u(x) = \int_\Omega G(x,y)g(y)dA(y) + \int_\Gamma H(x,y)h(y)ds(y) = [\mathbf{G}g](x) + [\mathbf{H}h](x).$$

Note that G and H are almost never known analytically, but one can prove that they exist and that they are in many ways "nice" away from the diagonal.

In the numerical PDE community, methods for constructing an approximation to the "solution operator" are known as *direct methods.*

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.

“Iterative” versus ”direct” solvers

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

$$A\,x = b.$$

Iterative methods:

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

Construct a sequence of vectors $x_1, x_2, x_3, \dots$ that (hopefully!) converge to the exact solution.

Many iterative methods access 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 A.)

(Exact except for rounding errors.)

Definition of a “fast direct solver”:

Given a system matrix A (often defined implicitly) and a computational tolerance ε, a “fast direct solver” constructs an operator T such that

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

Then an approximate solution to the solution to the linear system

$$A\,x = b$$

is obtained by simply evaluating

$$x = T\,b.$$

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

We say that the scheme is “fast” if it requires $O(N)$ operations to approximately invert an $N \times N$ matrix. (Complexity $O(N\,\log N)$, $O(N\,(\log N)^2)$, ... also OK.)

(Variation: Find factors B and C such that $||A - B\,C|| \leq \varepsilon$, and linear solves involving the matrices B and C are fast.)

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.)

2. Problems that are relatively ill-conditioned:
   - Scattering problems at intermediate or high 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.

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 should be viewed as a step towards getting a LAPACK-type environment for solving the basic linear boundary value problems of mathematical physics.

Origins of fast direct solvers:

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

1993? Fast inversion of 1D operators *Rokhlin and 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, et al,*

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

2002 inversion of "Hierarchically semi-separable" matrices, *M. Gu, S. Chandrasekharan, et al.*

2007 factorization of discrete Laplace operators, *S. Chandrasekharan, M. Gu, X.S. Li, J. Xia.*

Current status — problems with non-oscillatory kernels (Laplace, elasticity, 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:* Some $O(N\,(\log N)^p)$ inversion algorithms exist. Work remains — general grids, improve constants, *etc.*
- *Volume Int. Eq. in the plane (e.g. low frequency Lippman-Schwinger):* $O(N\,(\log N)^p)$ inversion algorithms exist. Implementation is under way.
- *Boundary Integral Equations in $\mathbb{R}^3$:* $O(N^{3/2})$ techniques exist. $O(N)$ techniques should be possible — but this is speculation at this point.

Problems on 3D domains:

- Exist, but currently involve large constants — improvements are imperative.

Current status — problems with oscillatory kernels (Helmholtz, Maxwell, etc).

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:* ???
  ($O(N^{3/2})$ inversion is possible.)
- *Volume Int. Eq. in the plane (e.g. high frequency Lippman-Schwinger):* ???
  ($O(N^{3/2})$ inversion is possible.)
- *Boundary Integral Equations in $\mathbb{R}^3$:* ???

Problems on 3D domains:

- ????
  ($O(N^2)$ inversion is possible.)

How do these algorithms actually work?

Let us consider the simplest case: fast inversion of an equation on a 1D domain.

Things are *still* very technical ... quite involved notation ...

What follows is a brief description of a method from an extreme birds-eye view.

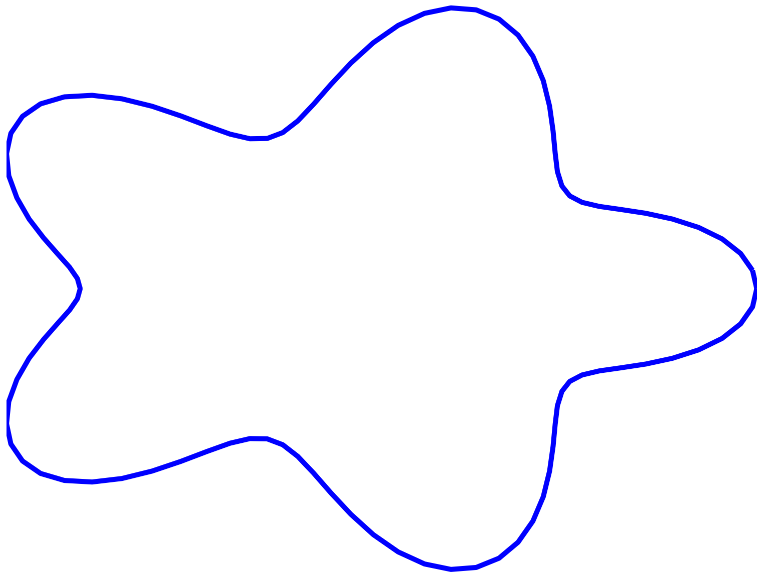
We start by describing some key properties of the matrices under consideration.

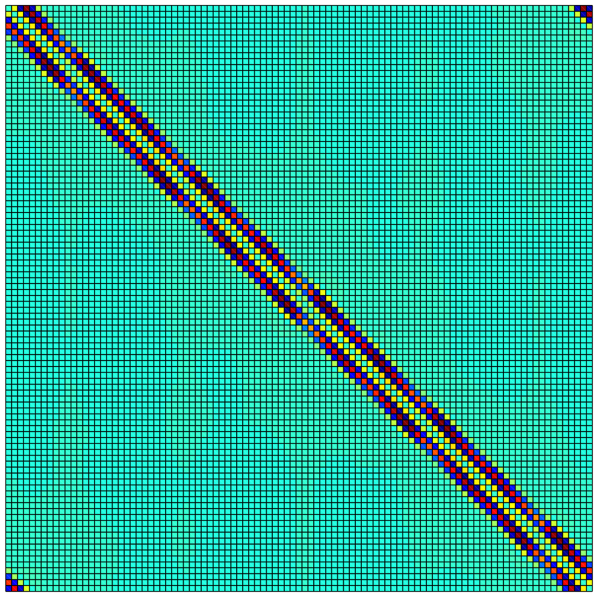
For concreteness, consider a 100×100 matrix A approximating the operator

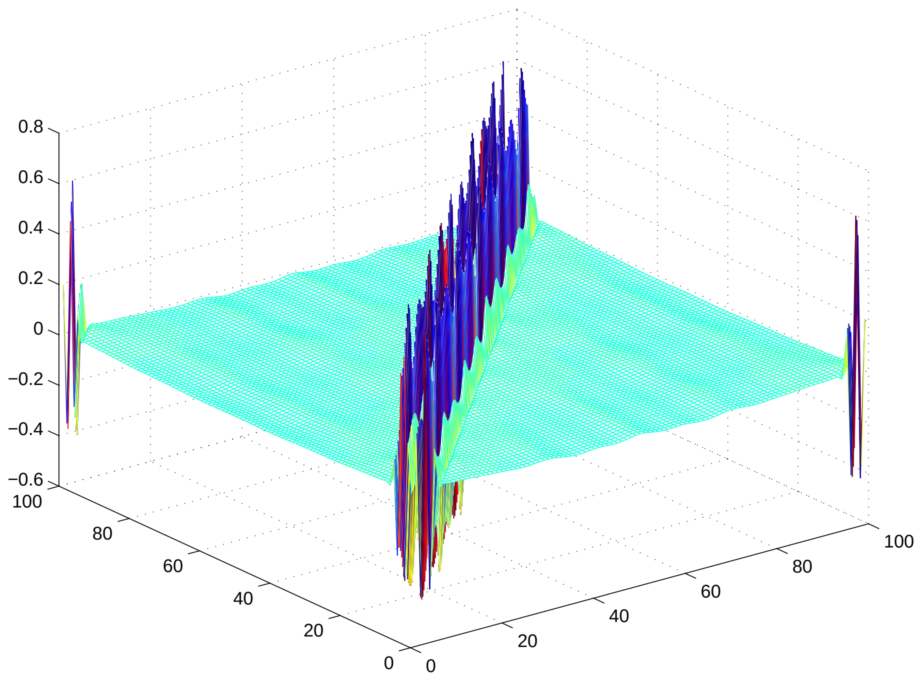
$$[\mathcal{S}_\Gamma\, u](x) = u(x) + \int_\Gamma \log|x-y|\, u(y)\, ds(y).$$

The matrix A is characterized by:

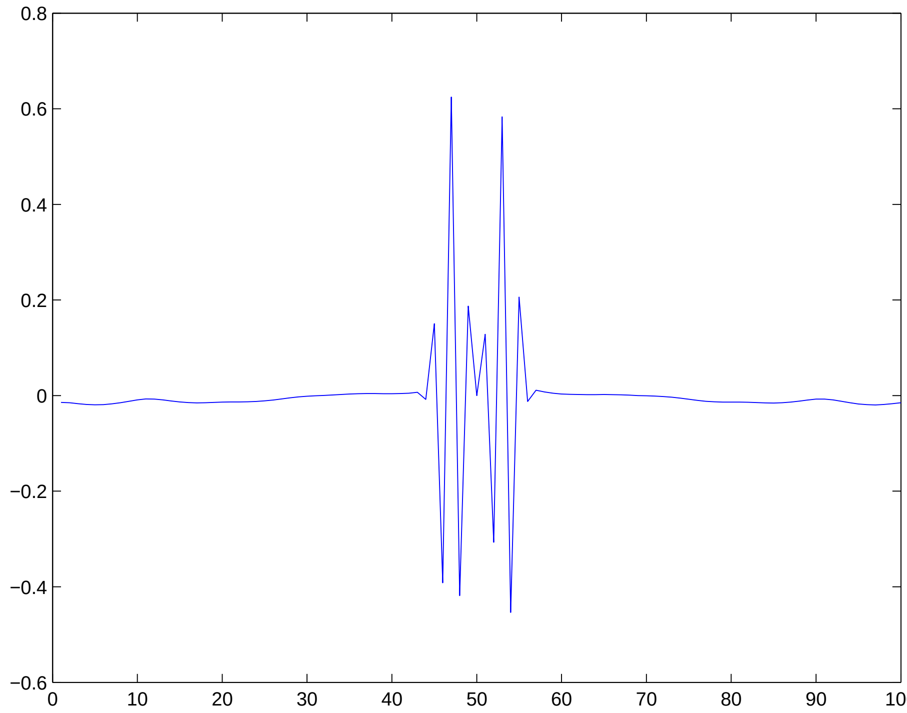
- Irregular behavior near the diagonal.
- Smooth entries away from the diagonal.


The contour Γ.

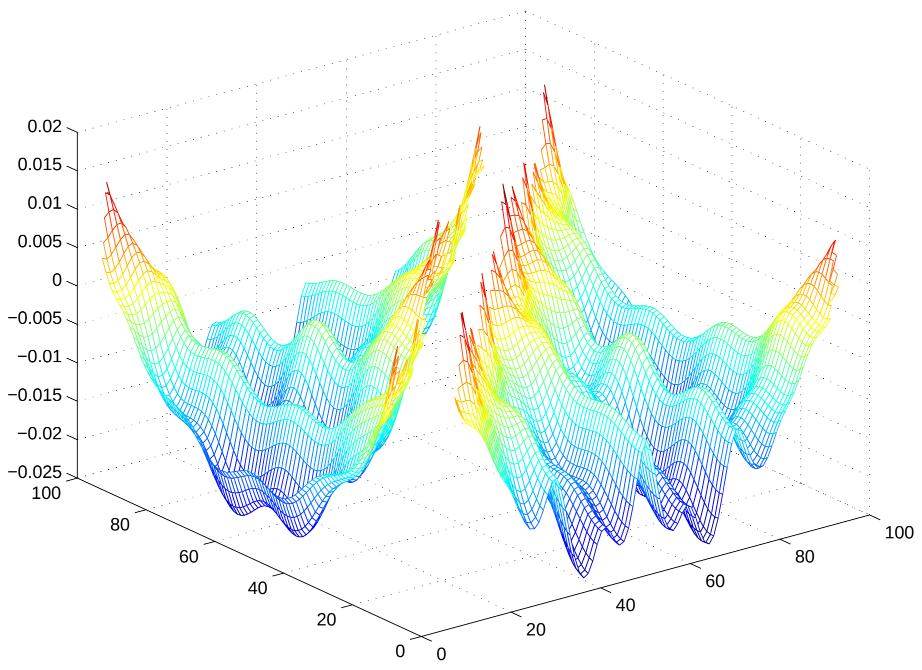

The matrix A.



Plot of A_{ij} vs i and j

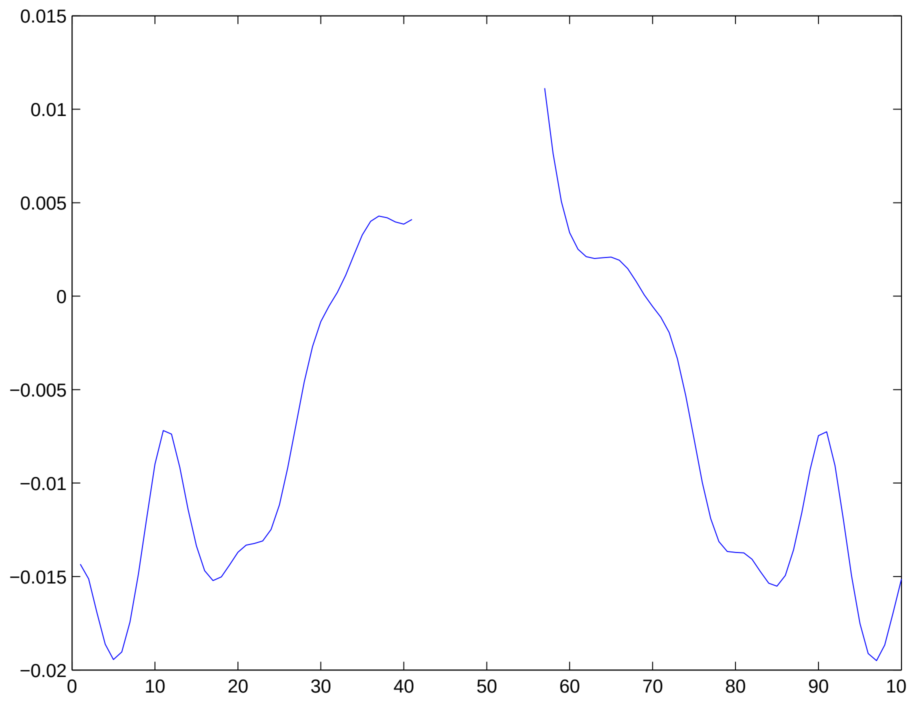


The 50$^{\text{th}}$ row of A



Plot of A_{ij} vs i and j

(without the diagonal entries)

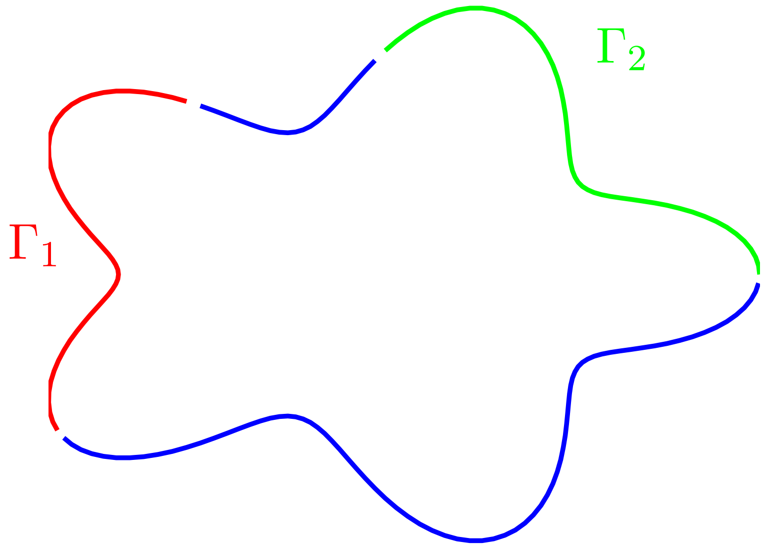


The 50th row of A

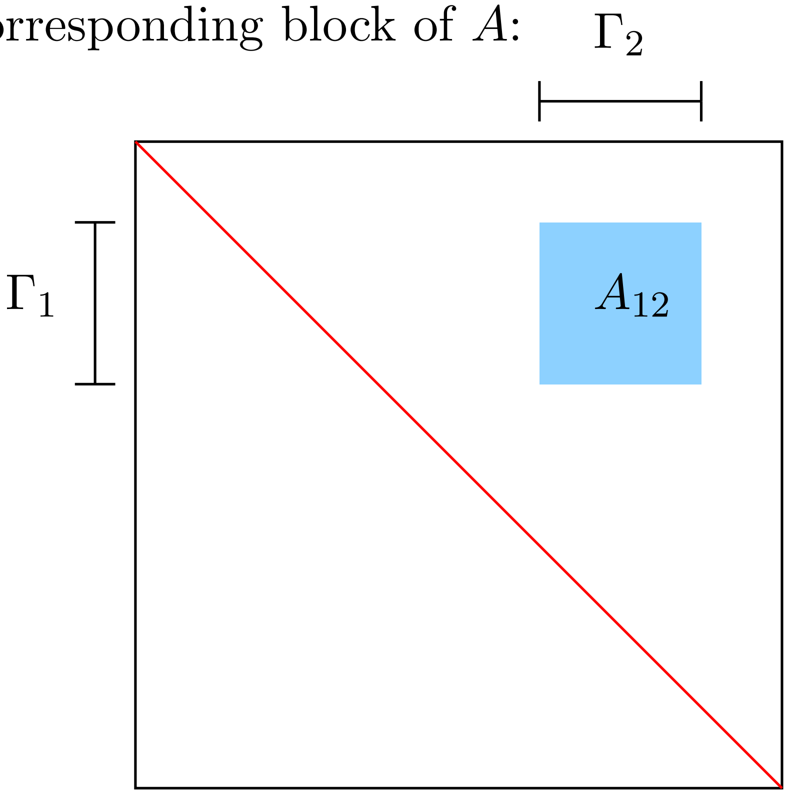
(without the diagonal entries)

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

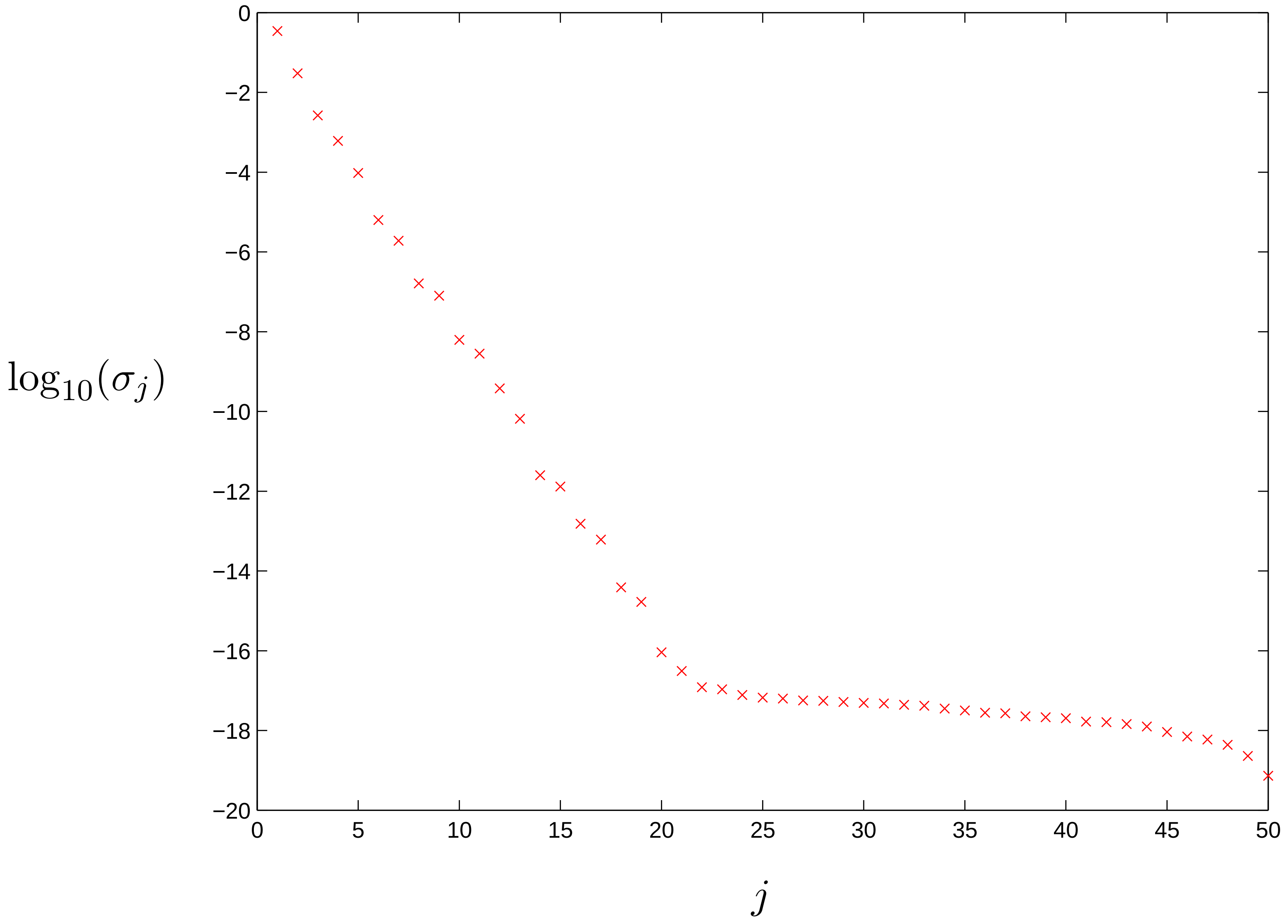


The matrix A

The block A_{12} is a discretization of the integral operator

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

Singular values of A_{12} (now for a 200×200 matrix A):



Let A be a matrix consisting of $p \times p$ blocks of size $n \times n$:

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

Core assumption: Each off-diagonal block A_{ij} admits the factorization

$$\begin{array}{ccccc} A_{ij} & = & U_i & \tilde{A}_{ij} & V_j^* \\ n \times n & & n \times k & k \times k & k \times n \end{array}$$

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 U_i for their column spaces (and analogously all blocks in the j'th column use the same basis matrices V_j for their row spaces).

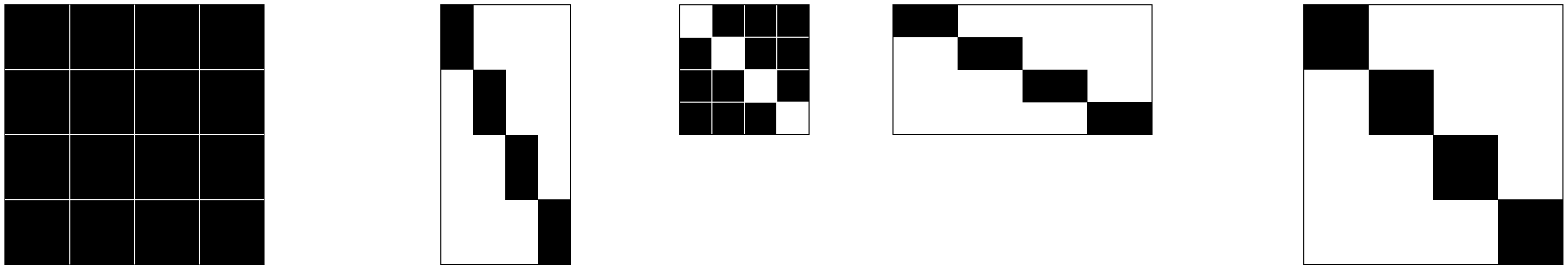
We get $A = \begin{bmatrix} D_{11} & U_1\,\tilde{A}_{12}\,V_2^* & U_1\,\tilde{A}_{13}\,V_3^* & U_1\,\tilde{A}_{14}\,V_4^* \\ U_2\,\tilde{A}_{21}\,V_1^* & D_{22} & U_2\,\tilde{A}_{23}\,V_3^* & U_2\,\tilde{A}_{24}\,V_4^* \\ U_3\,\tilde{A}_{31}\,V_1^* & U_3\,\tilde{A}_{32}\,V_2^* & D_{33} & U_3\,\tilde{A}_{34}\,V_4^* \\ U_4\,\tilde{A}_{41}\,V_1^* & U_4\,\tilde{A}_{42}\,V_2^* & U_4\,\tilde{A}_{43}\,V_3^* & D_{44} \end{bmatrix}$.

Then A admits the factorization:

$$A = \underbrace{\begin{bmatrix} U_1 & & & \\ & U_2 & & \\ & & U_3 & \\ & & & U_4 \end{bmatrix}}_{=U} \underbrace{\begin{bmatrix} 0 & \tilde{A}_{12} & \tilde{A}_{13} & \tilde{A}_{14} \\ \tilde{A}_{21} & 0 & \tilde{A}_{23} & \tilde{A}_{24} \\ \tilde{A}_{31} & \tilde{A}_{32} & 0 & \tilde{A}_{34} \\ \tilde{A}_{41} & \tilde{A}_{42} & \tilde{A}_{43} & 0 \end{bmatrix}}_{=\tilde{A}} \underbrace{\begin{bmatrix} V_1^* & & & \\ & V_2^* & & \\ & & V_3^* & \\ & & & V_4^* \end{bmatrix}}_{=V^*} + \underbrace{\begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & D_3 & \\ & & & D_4 \end{bmatrix}}_{=D}$$

or

$$\begin{array}{ccccccc} A & = & U & \tilde{A} & V^* & + & D, \\ pn \times pn & & pn \times pk & pk \times pk & pk \times pn & & pn \times pn \end{array}$$



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

$$\begin{array}{ccccccc} A & = & U & \tilde{A} & V^* & + & D, \\ N \times N & & N \times K & K \times K & K \times N & & N \times N \end{array}$$

then

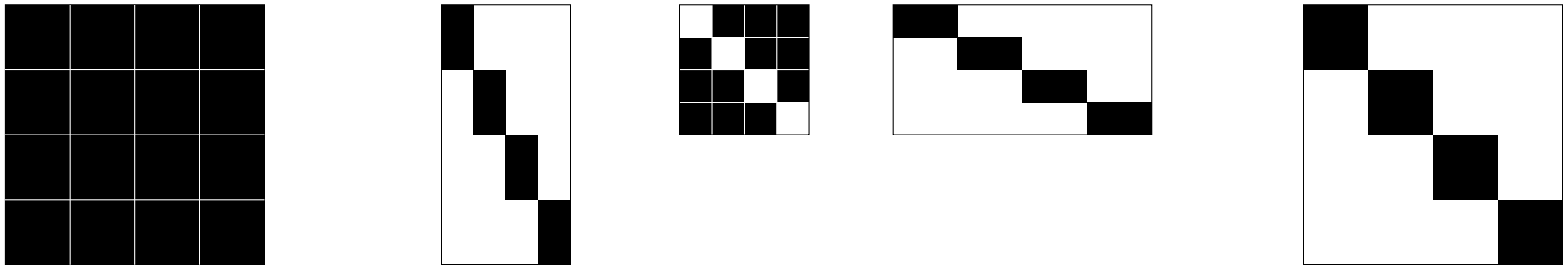
$$\begin{array}{ccccccc} A^{-1} & = & E & (\tilde{A} + \hat{D})^{-1} & F^* & + & G, \\ N \times N & & N \times K & K \times K & K \times N & & N \times N \end{array}$$

where (provided all intermediate matrices are invertible)

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

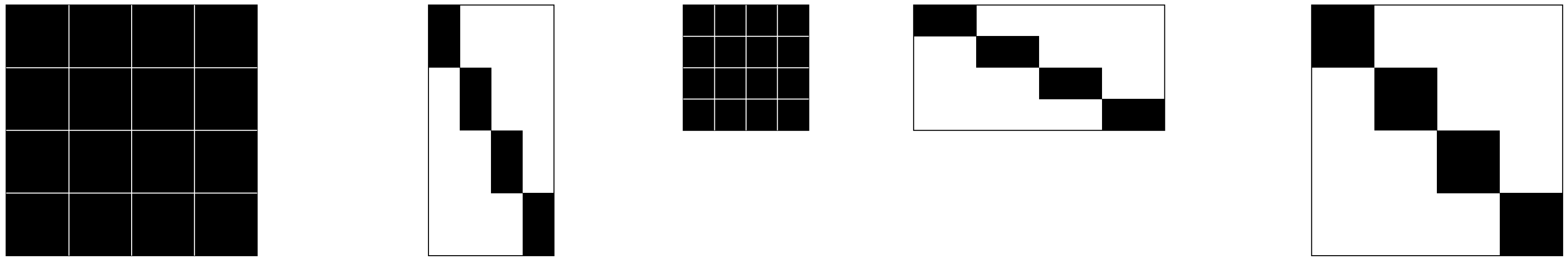
Lemma: [Variation of Woodbury] If an $N \times N$ matrix A admits the factorization

$$\begin{array}{ccccccc} A & = & U & \tilde{A} & V^* & + & D, \\ p\,n \times p\,n & & p\,n \times p\,k & p\,k \times p\,k & p\,k \times p\,n & & p\,n \times p\,n \end{array}$$



then

$$\begin{array}{ccccccc} A^{-1} & = & E & (\tilde{A} + \hat{D})^{-1} & F^* & + & G, \\ p\,n \times p\,n & & p\,n \times p\,k & p\,k \times p\,k & p\,k \times p\,n & & p\,n \times p\,n \end{array}$$



where (provided all intermediate matrices are invertible)

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

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: A has $p \times p$ blocks, each of size $n \times n$ and of rank k.)

We must recurse!

We must recurse!

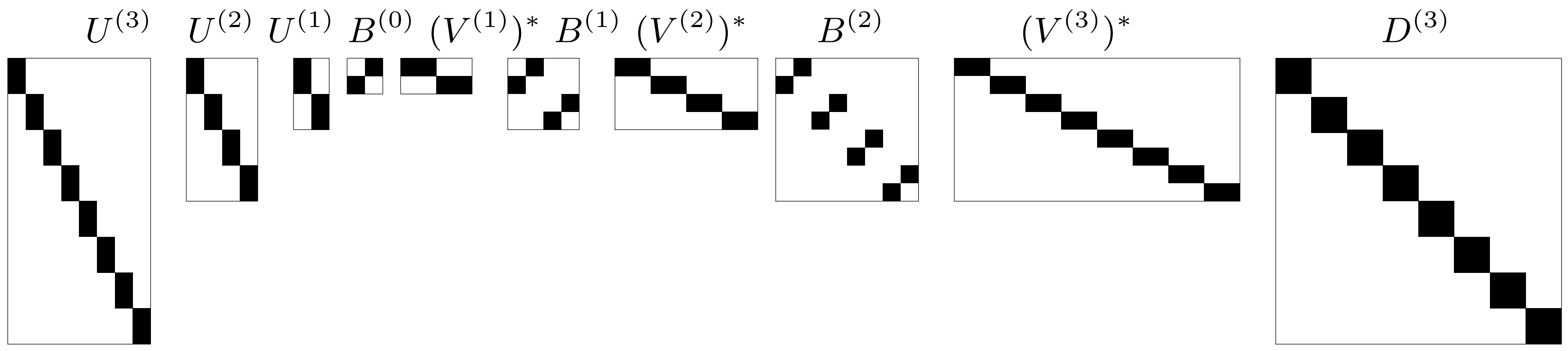
Using a telescoping factorization of A:

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

we have a formula

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

Block structure of factorization:



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

Many important details are left out.

Among the more important ones:

- How do you represent potentials?
  Multipole expansions, proxy charges, interpolation, ...

- How do you compute the telescoping factorization in the first place?

- Generalization to
  - Volume integral equations in $\mathbb{R}^2$.
  - Boundary integral equations in $\mathbb{R}^3$.

  (Volume integral equations in $\mathbb{R}^3$ are currently not practical.)

Numerical examples

The computational examples are assembled to illustrate the asymptotic scaling of the methods.

Most of the examples are old: They were generated around 2005 on a PC from 2002 (a 2.8Ghz machine with 512Mb of RAM).

Other examples are more recent, these were implemented in Matlab.

Example: An exterior Laplace Dirichlet problem

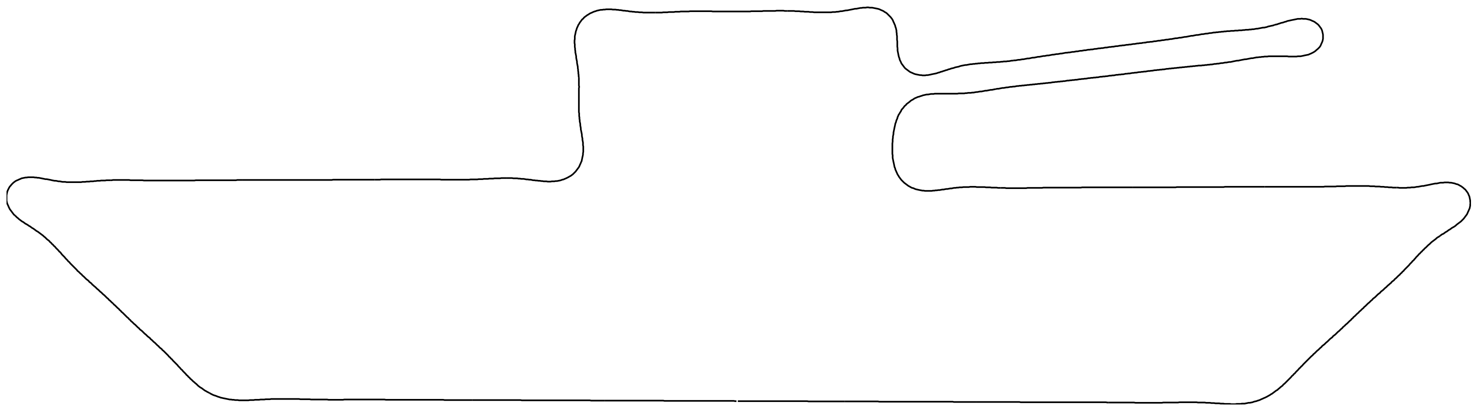
We invert a matrix approximating the operator

$$[A\,u](x) = \frac{1}{2}\,u(x) + \frac{1}{2\,\pi}\int_\Gamma D(x,y)\,u(y)\,ds(y).$$

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

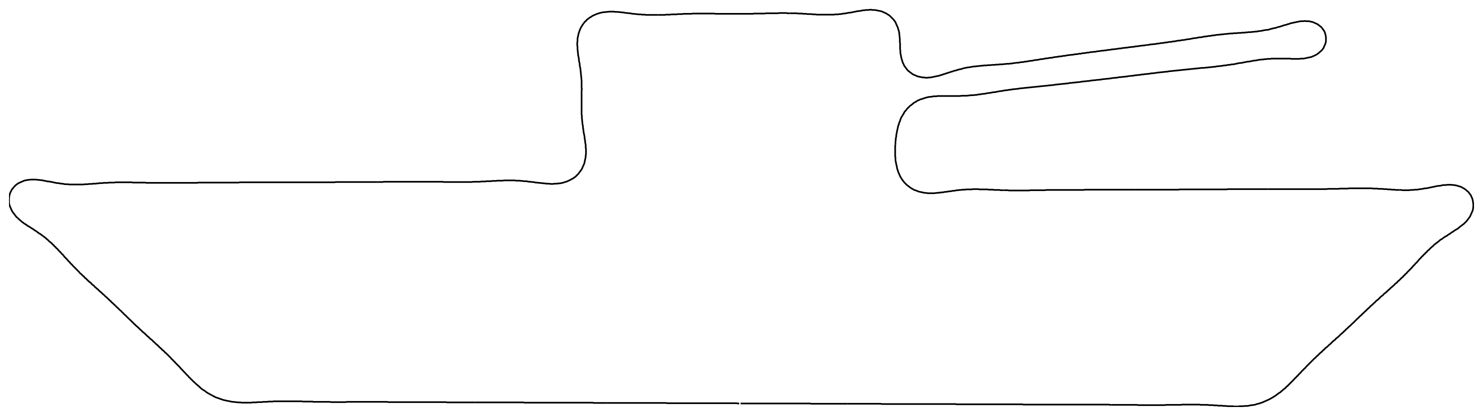
$$D(x,y) = -\frac{1}{2\pi}\,\frac{n(y)\cdot(x-y)}{|x-y|^2},$$

and where Γ is the countour:



N_{start}	N_{final}	t_{tot} (sec)	t_{solve} (sec)	E_{res}	E_{pot}	$\sigma_{\min}$	Memory (Mb)
400	301	5.3e-01	2.9e-03	4.7e-10	3.0e-06	1.3e-02	4.2e+00
800	351	9.6e-01	4.1e-03	2.2e-10	6.3e-10	1.2e-02	6.5e+00
1600	391	1.6e+00	6.3e-03	1.3e-10	1.6e-10	1.2e-02	9.2e+00
3200	391	1.8e+00	8.5e-03	6.6e-11	3.7e-10	1.2e-02	1.1e+01
6400	391	2.2e+00	1.2e-02	5.9e-11	8.9e-11	1.2e-02	1.4e+01
12800	390	2.6e+00	1.9e-02	3.6e-11	5.9e-11	1.2e-02	2.1e+01
25600	391	3.9e+00	3.4e-02	2.7e-11	4.7e-10	—	3.5e+01
51200	393	6.5e+00	6.5e-02	2.5e-11	5.3e-11	—	6.3e+01
102400	402	1.3e+01	1.2e-01	2.0e-11	—	—	1.2e+02

Example: An exterior Helmholtz Dirichlet problem



A smooth contour. Its length is roughly 15 and its horizontal width is 2.

The “combined field formulation” is used in forming the BIE.

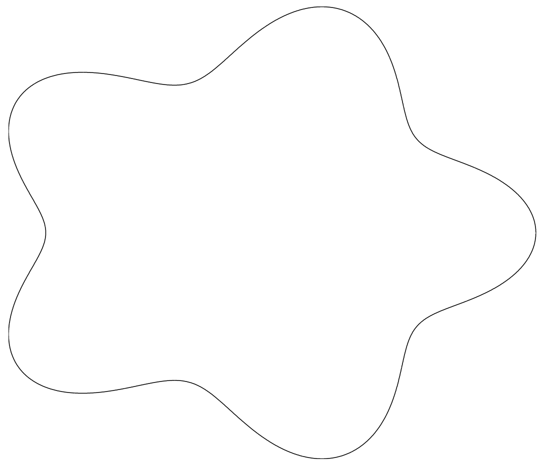
k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{res}	E_{pot}	σ_{min}	M
21	800	435	1.5e+01	3.3e-02	9.7e-08	7.1e-07	6.5e-01	12758
40	1600	550	3.0e+01	6.7e-02	6.2e-08	4.0e-08	8.0e-01	25372
79	3200	683	5.3e+01	1.2e-01	5.3e-08	3.8e-08	3.4e-01	44993
158	6400	870	9.2e+01	2.0e-01	3.9e-08	2.9e-08	3.4e-01	81679
316	12800	1179	1.8e+02	3.9e-01	2.3e-08	2.0e-08	3.4e-01	160493
632	25600	1753	4.3e+02	8.0e-01	1.7e-08	1.4e-08	3.3e-01	350984

Computational results for an exterior Helmholtz Dirichlet problem discretized with 10th order accurate quadrature. The Helmholtz parameter was chosen to keep the number of discretization points per wavelength constant at roughly 45 points per wavelength (resulting in a quadrature error about 10^{-12}).

Eventually ... the complexity is $O(n + k^3)$.

(Corresponding Laplace problems are *much* faster,
inversion of a $10^5 \times 10^5$ matrix takes less than 20 seconds.)

Example: An interior Helmholtz Dirichlet problem

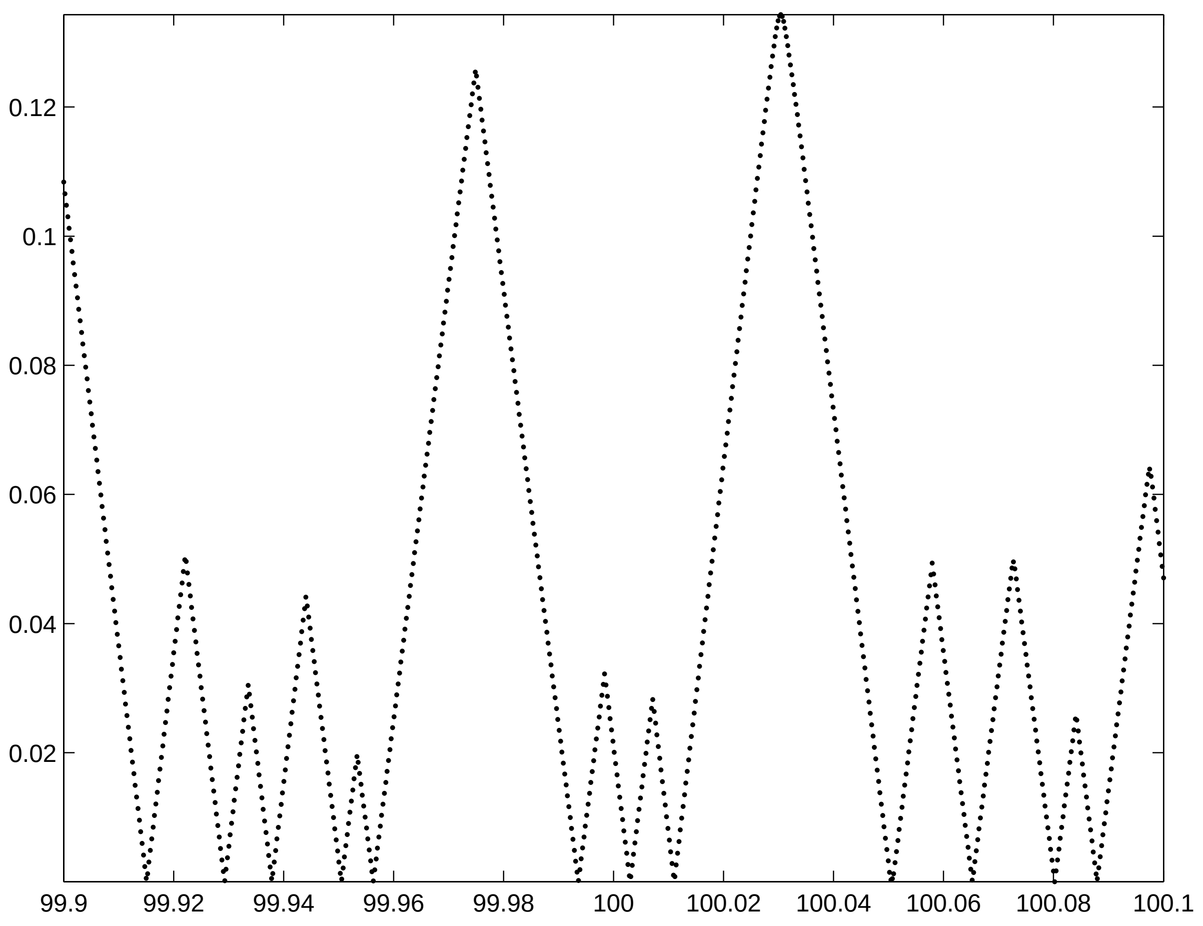


The diameter of the contour is about 2.5. An interior Helmholtz problem with Dirichlet boundary data was solved using $N = 6\,400$ discretization points, with a prescribed accuracy of 10^{-10}.

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

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.

What about finite element matrices?

These look quite different — *very* large, sparse, ...

However, their *inverses* have the rank structure of discretized integral operators.

Example: Consider the BVP

$$\begin{cases} -\Delta\, u(x) + b(x)\cdot\nabla u(x) + c(x)\, u(x) = g(x), & x \in \Omega, \\ \dfrac{\partial u(x)}{\partial n} = h, & x \in \Gamma. \end{cases}$$

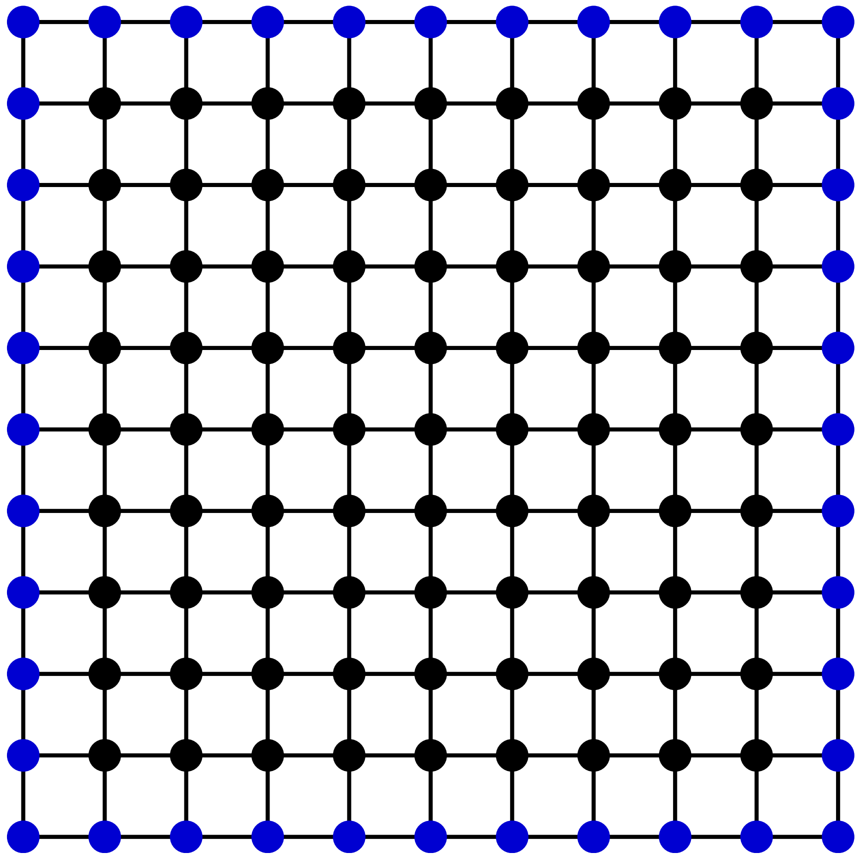
The finite element method produces a large sparse matrix A whose action mimics the action of the differential operator.

The *inverse of A* mimics the action of the solution operator

$$u(x) = \int_\Omega G(x,y)\, g(y)\, dA(y) + \int_\Gamma H(x,y)\, h(y)\, ds(y).$$

where G is the Green's function of the problem, and H is the N-to-D operator.

Example: Inversion of a “Finite Element Matrix”



A grid conduction problem (the “five-point stencil”).

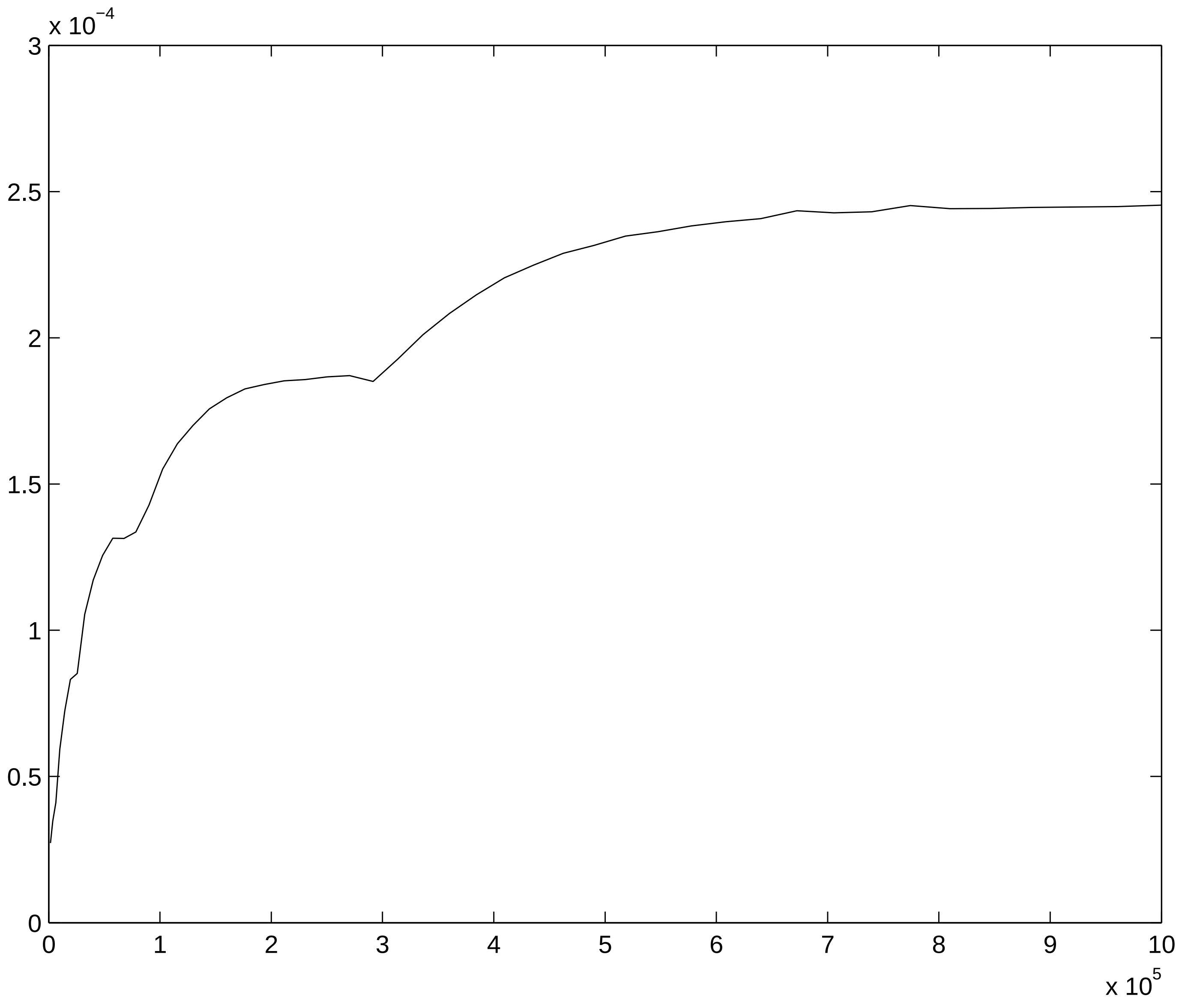
The conductivity of each bar is a random number drawn from a uniform distribution on [1, 2].

If all conductivities were one, then we would get the standard five-point stencil:

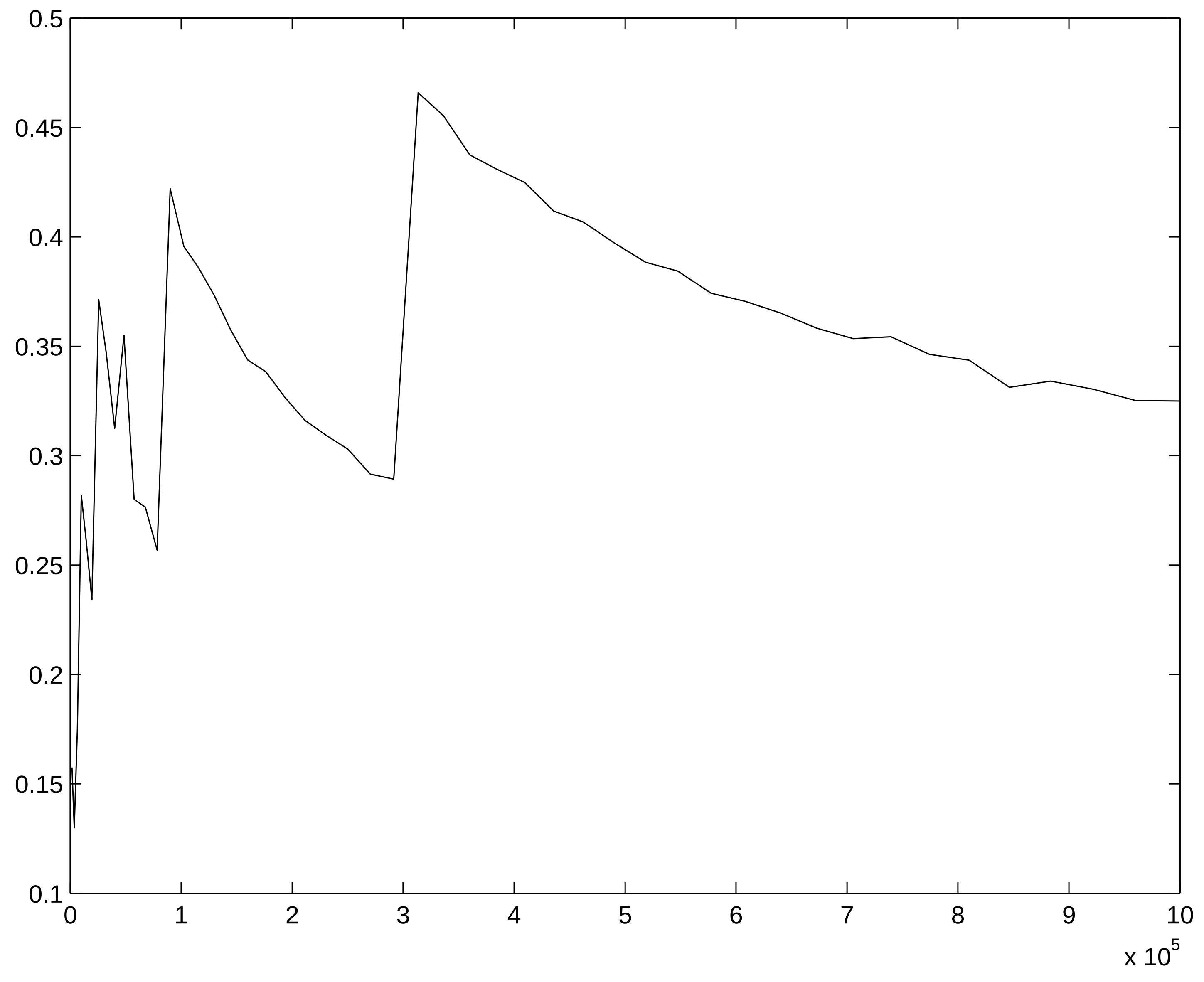
$$A = \begin{bmatrix} C & -I & 0 & 0 & \cdots \\ -I & C & -I & 0 & \cdots \\ 0 & -I & C & -I & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \qquad C = \begin{bmatrix} 4 & -1 & 0 & 0 & \cdots \\ -1 & 4 & -1 & 0 & \cdots \\ 0 & -1 & 4 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}.$$

N	T_{solve} (seconds)	T_{apply} (seconds)	M (kB)	e_1	e_2	e_3	e_4
10 000	5.93e-1	2.82e-3	3.82e+2	1.29e-8	1.37e-7	2.61e-8	3.31e-8
40 000	4.69e+0	6.25e-3	9.19e+2	9.35e-9	8.74e-8	4.71e-8	6.47e-8
90 000	1.28e+1	1.27e-2	1.51e+3	—	—	7.98e-8	1.25e-7
160 000	2.87e+1	1.38e-2	2.15e+3	—	—	9.02e-8	1.84e-7
250 000	4.67e+1	1.52e-2	2.80e+3	—	—	1.02e-7	1.14e-7
360 000	7.50e+1	2.62e-2	3.55e+3	—	—	1.37e-7	1.57e-7
490 000	1.13e+2	2.78e-2	4.22e+3	—	—	—	—
640 000	1.54e+2	2.92e-2	5.45e+3	—	—	—	—
810 000	1.98e+2	3.09e-2	5.86e+3	—	—	—	—
1 000 000	2.45e+2	3.25e-2	6.66e+3	—	—	—	—

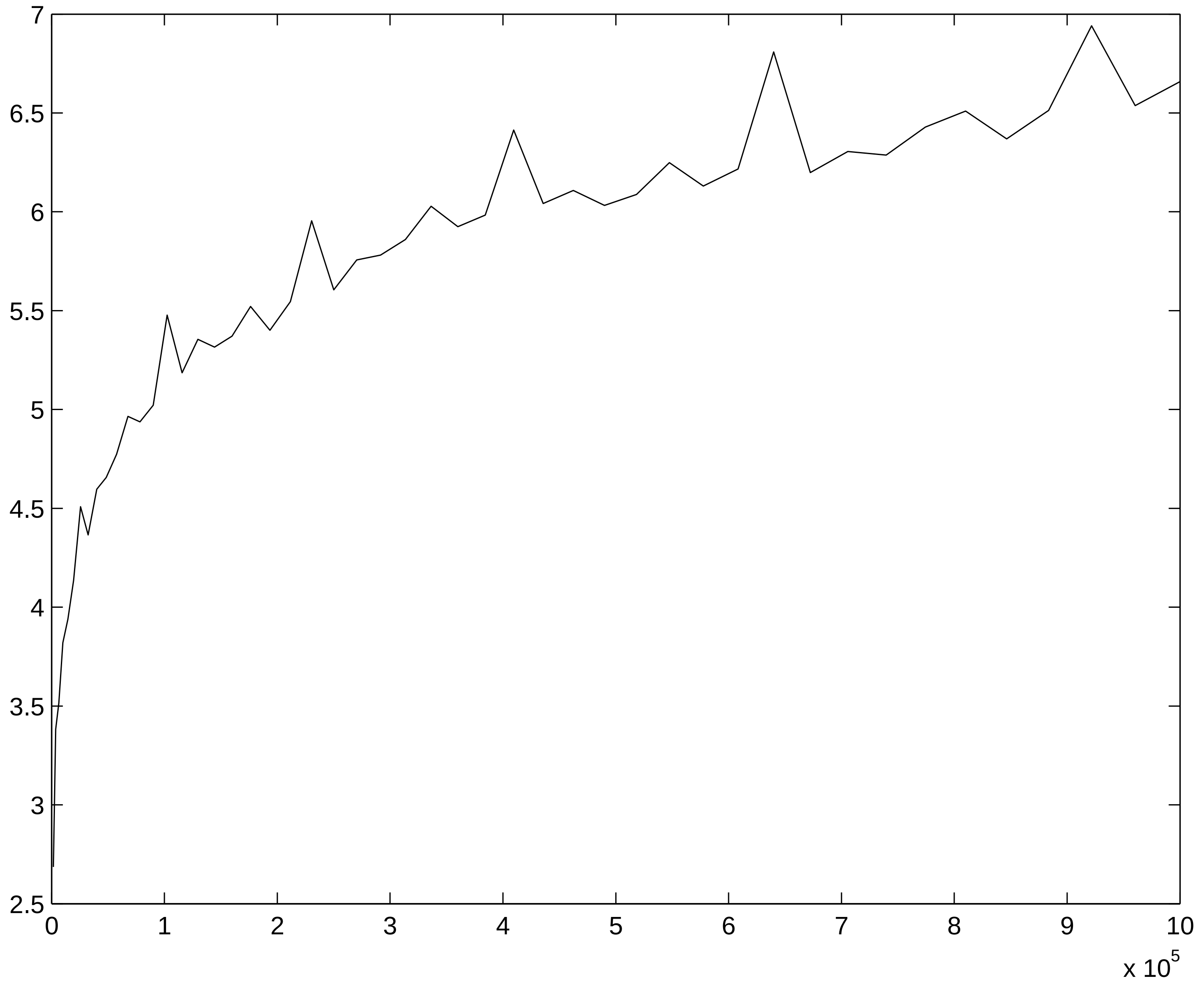
T_{apply}	Time required to apply a Dirichlet-to-Neumann op. (of size $4\sqrt{N} \times 4\sqrt{N}$)
e_1	The largest error in any entry of $\tilde{A}_n^{-1}$
e_2	The error in l^2-operator norm of $\tilde{A}_n^{-1}$
e_3	The l^2-error in the vector $\tilde{A}_{nn}^{-1}\, r$ where r is a unit vector of random direction.
e_4	The l^2-error in the first column of $\tilde{A}_{nn}^{-1}$.



$$\frac{T_{\text{invert}}}{N} \text{ versus } \quad N$$



$$\frac{T_{\text{apply}}}{\sqrt{N}} \quad \text{versus} \quad N$$



$$\frac{M}{\sqrt{N}} \text{ versus } N.$$

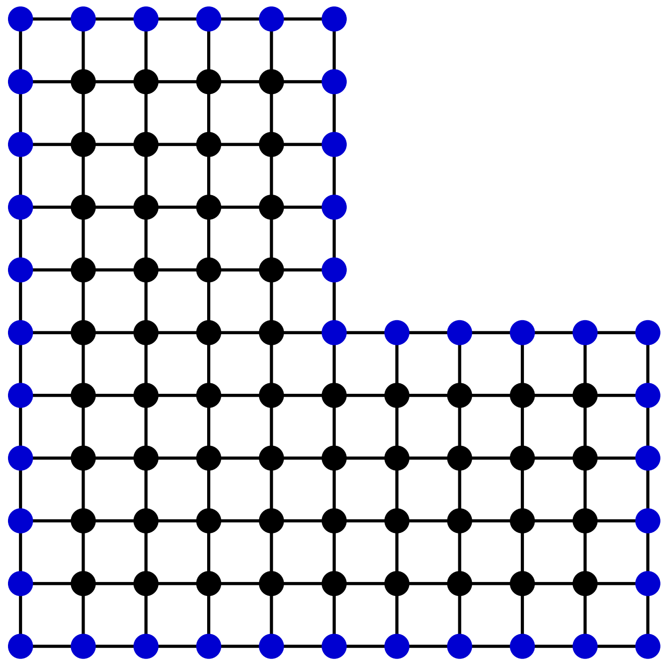
Example: A lattice domain.

Let Ω denote a subset of $\mathbb{Z}^2$ (black nodes) with boundary Γ (blue nodes). Then consider the Neumann problem

$$\begin{aligned} Au(m) &= 0, && m \in \Omega, \\ \partial_\nu u(m) &= h(m), && m \in \Gamma, \end{aligned}$$

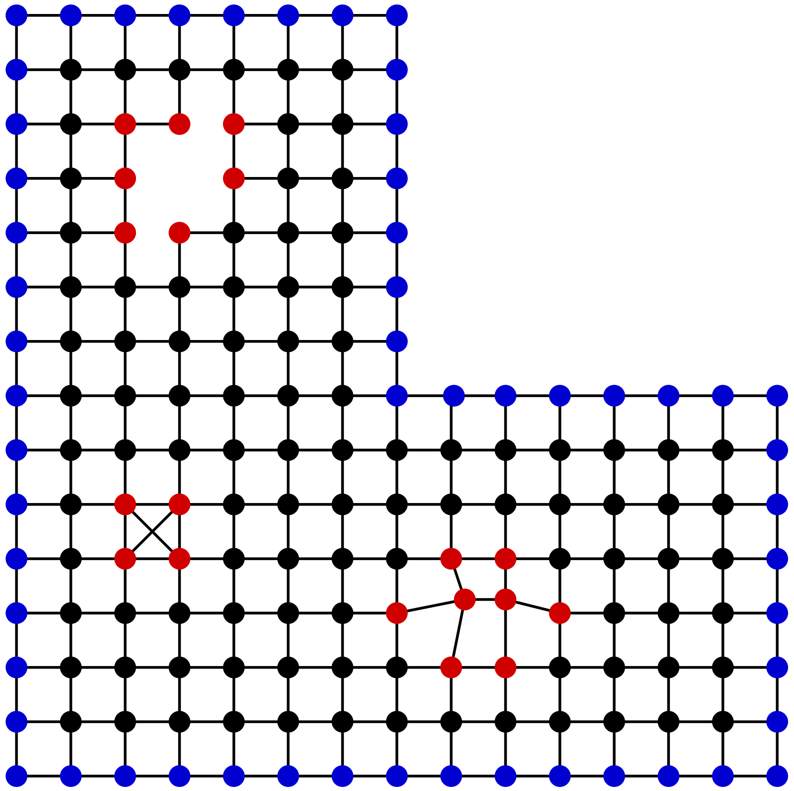
where A is the five-point stencil

$$\begin{aligned} Au(m) = 4u(m) - u(m-e_1) - u(m+e_1) \\ - u(m-e_2) - u(m+e_2). \end{aligned}$$



By exploiting the existence of a *lattice fundamental solution*, the Neumann-to-Dirichlet operator can be constructed in $O(N_\Gamma)$ operations where N_Γ is the number of points on the boundary.

For a lattice with $N_\Gamma = 100\,000$ nodes on the boundary (and $7\,500\,000\,000$ interior nodes), the construction takes about 100 seconds at an accuracy of 10^{-7} when implemented in Matlab and executed on a standard desktop PC. Application of the N-to-D operator takes 1 second.



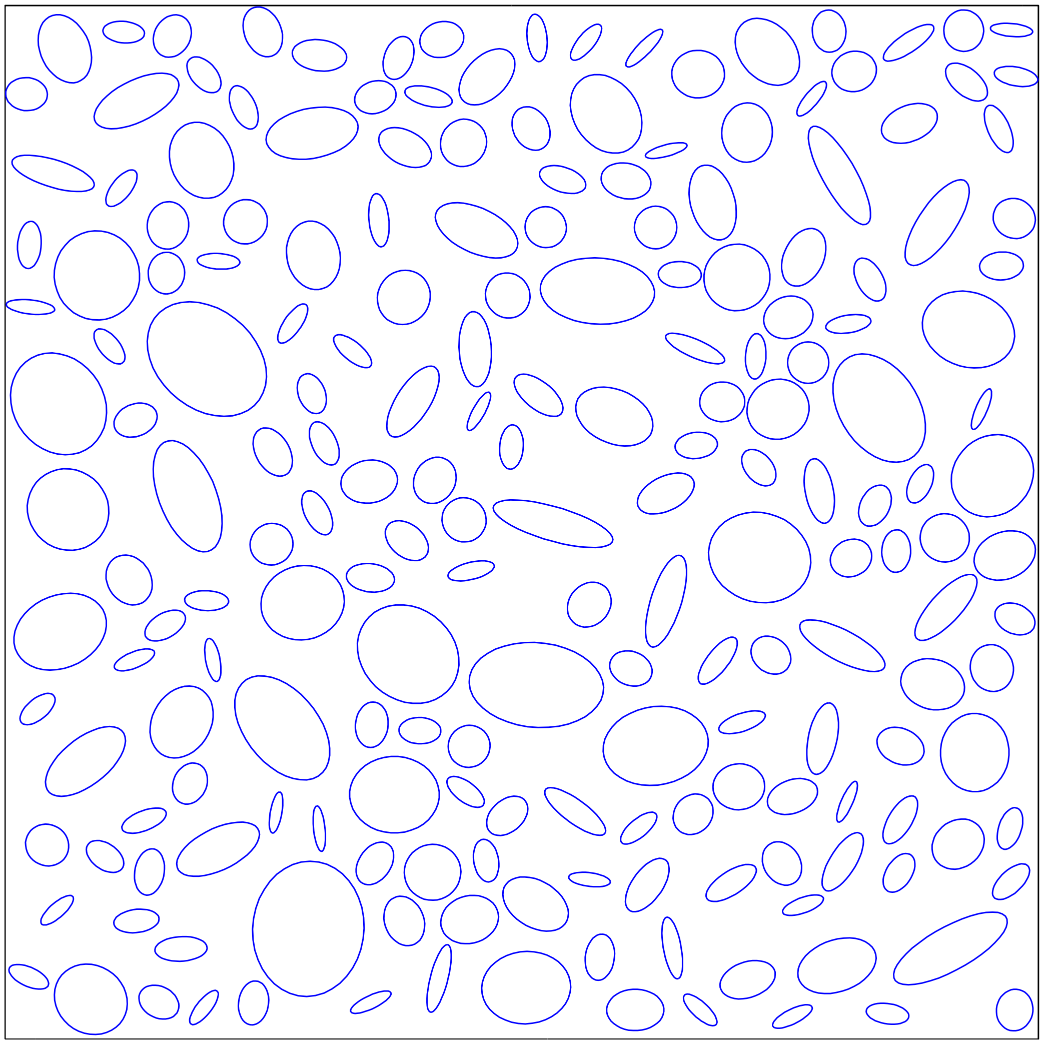
For a lattice with inclusions, the cost of constructing the N-to-D operator is

$$O(N_\Gamma + N_{\text{inc}}),$$

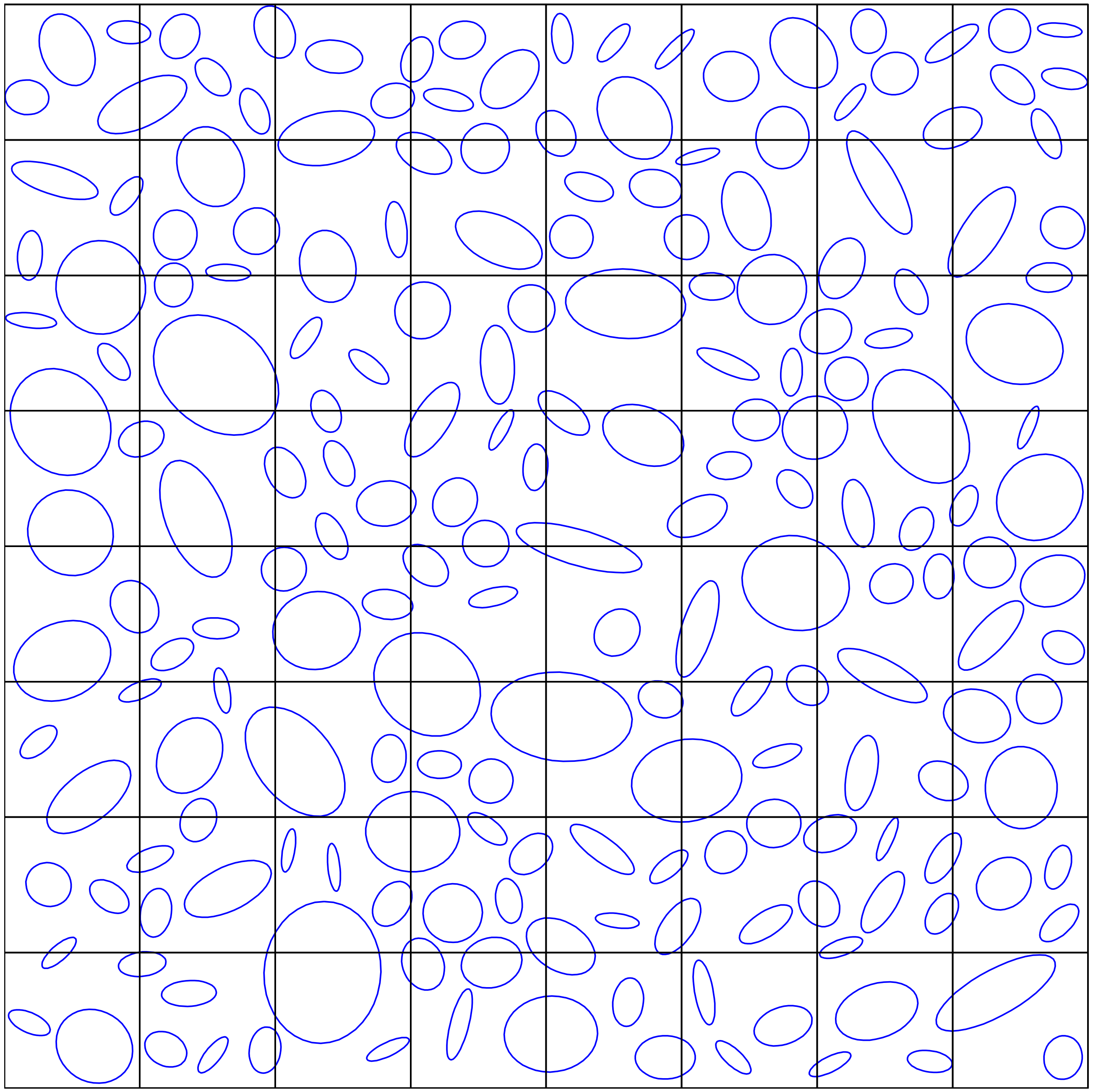
where N_Γ is the number of points on the boundary, and N_{inc} is the number of points at which periodicity is violated.

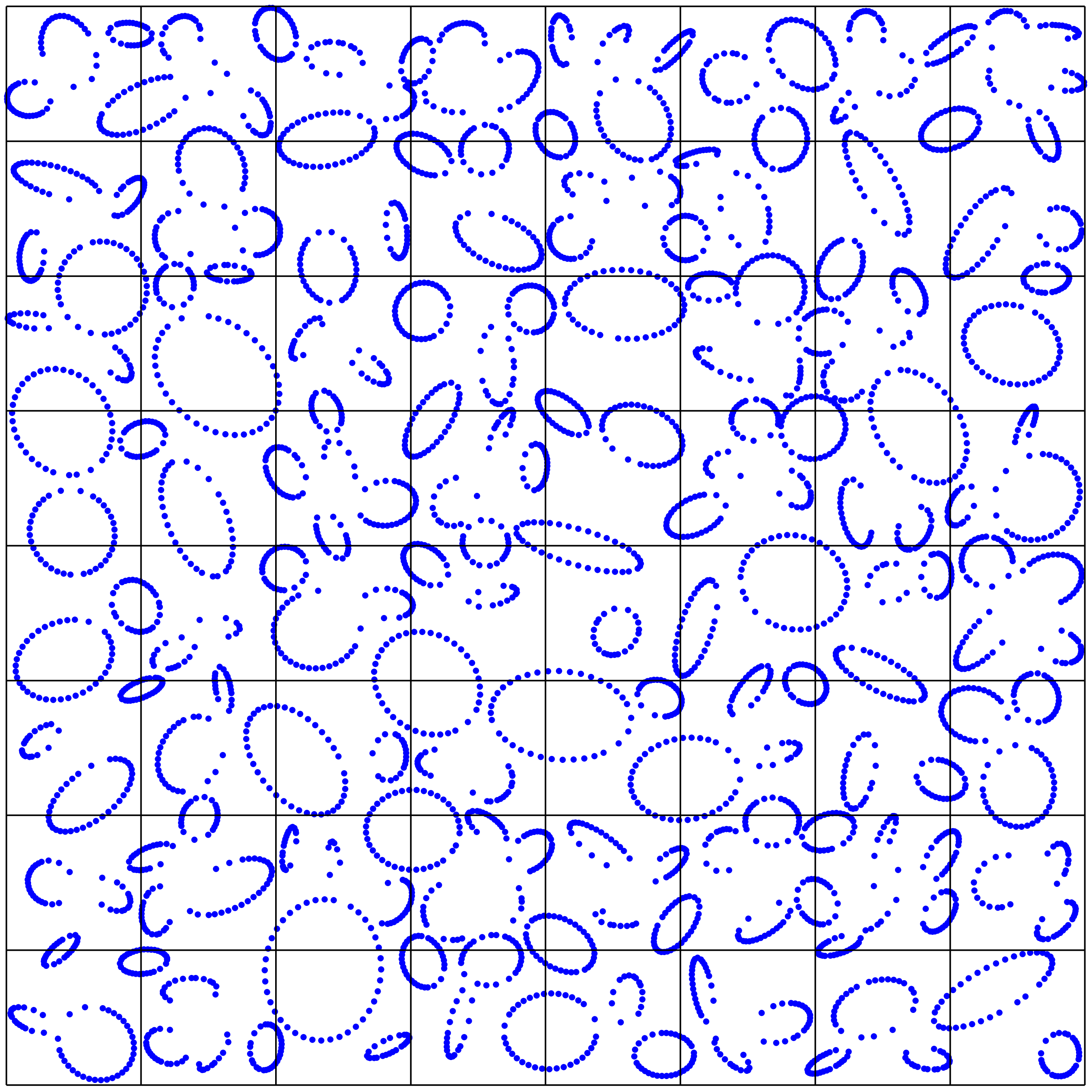
Extension to the case of continuum periodic micro-structure is under way.

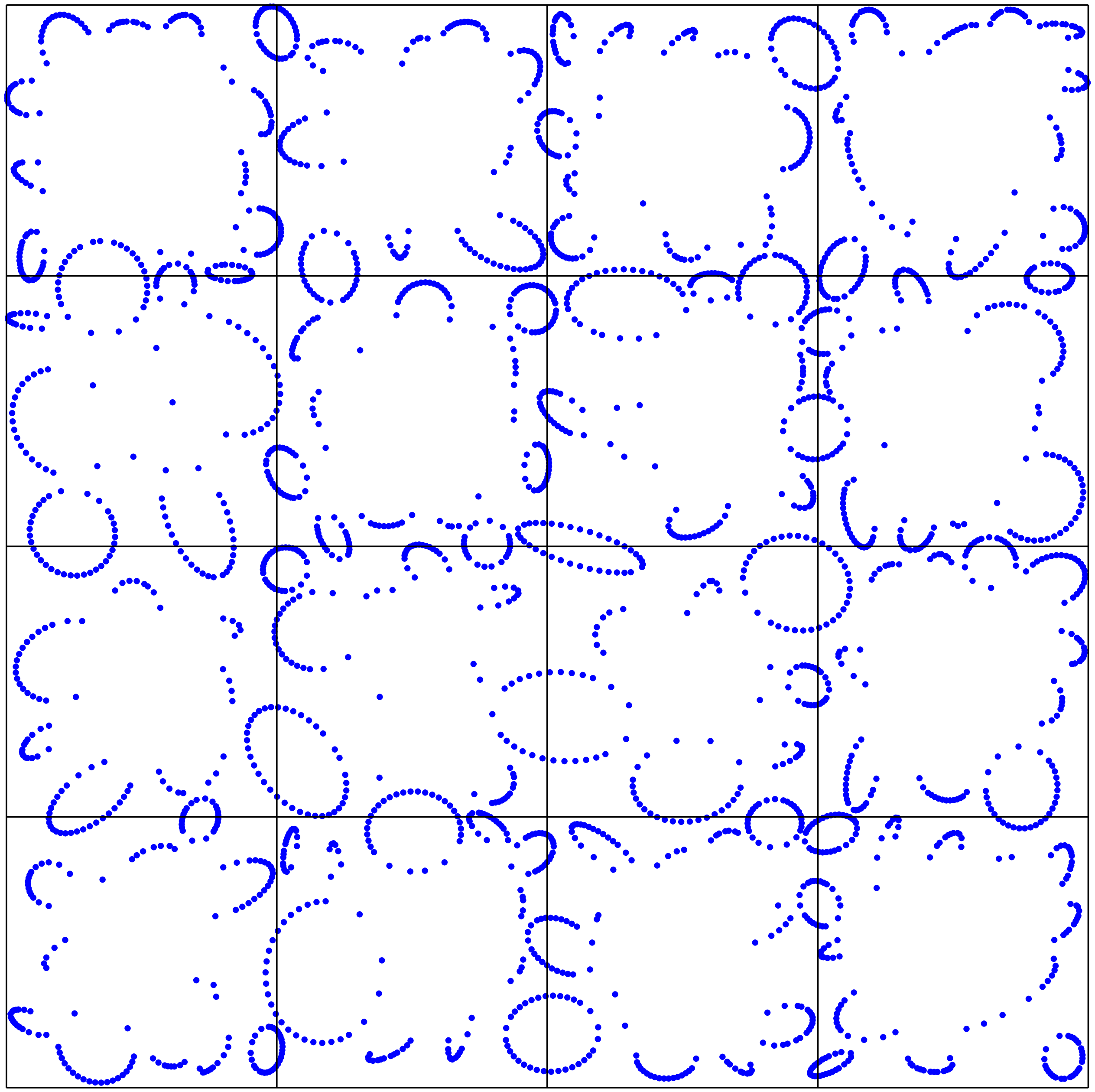
Example: A two-phase domain.

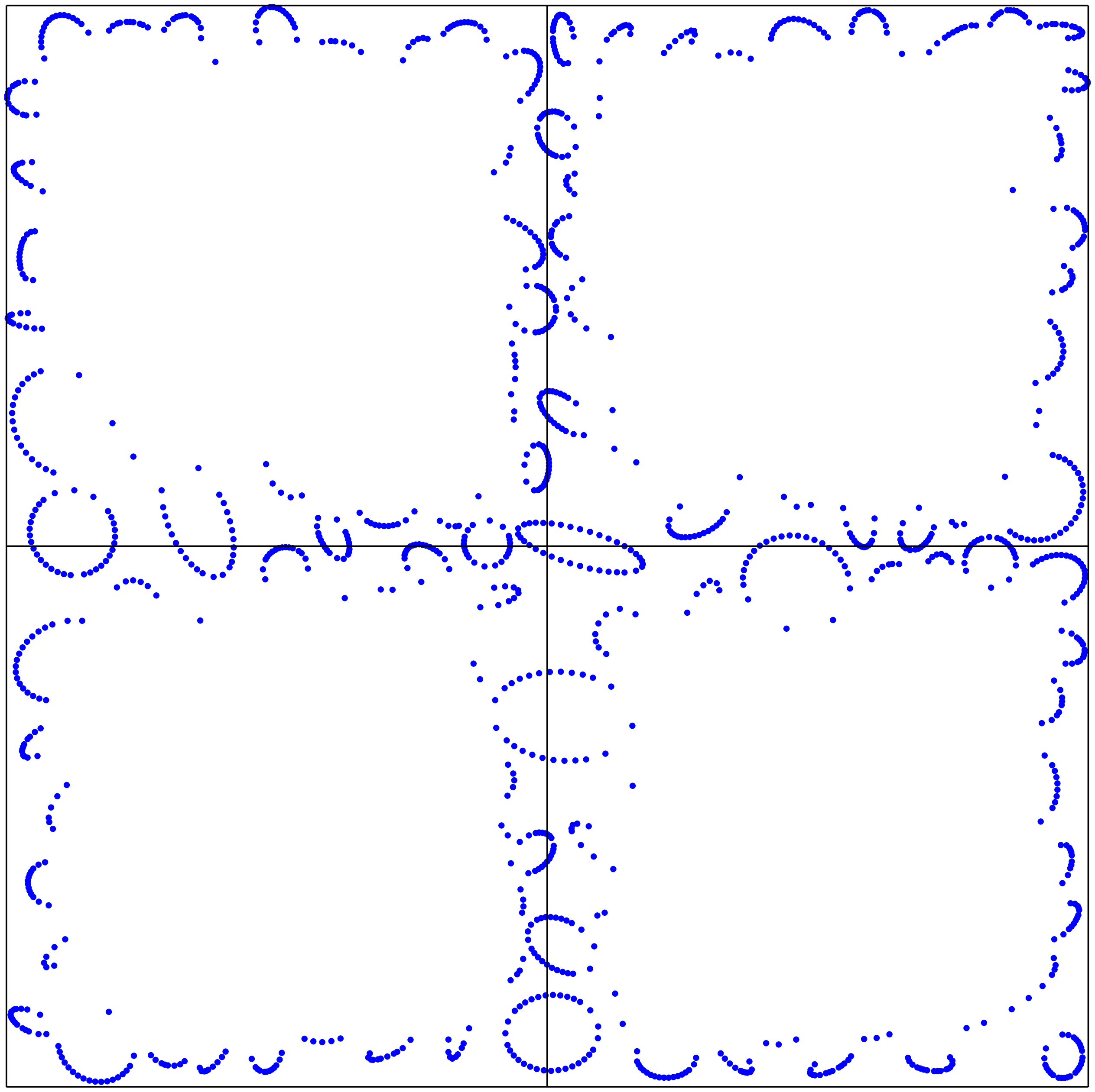


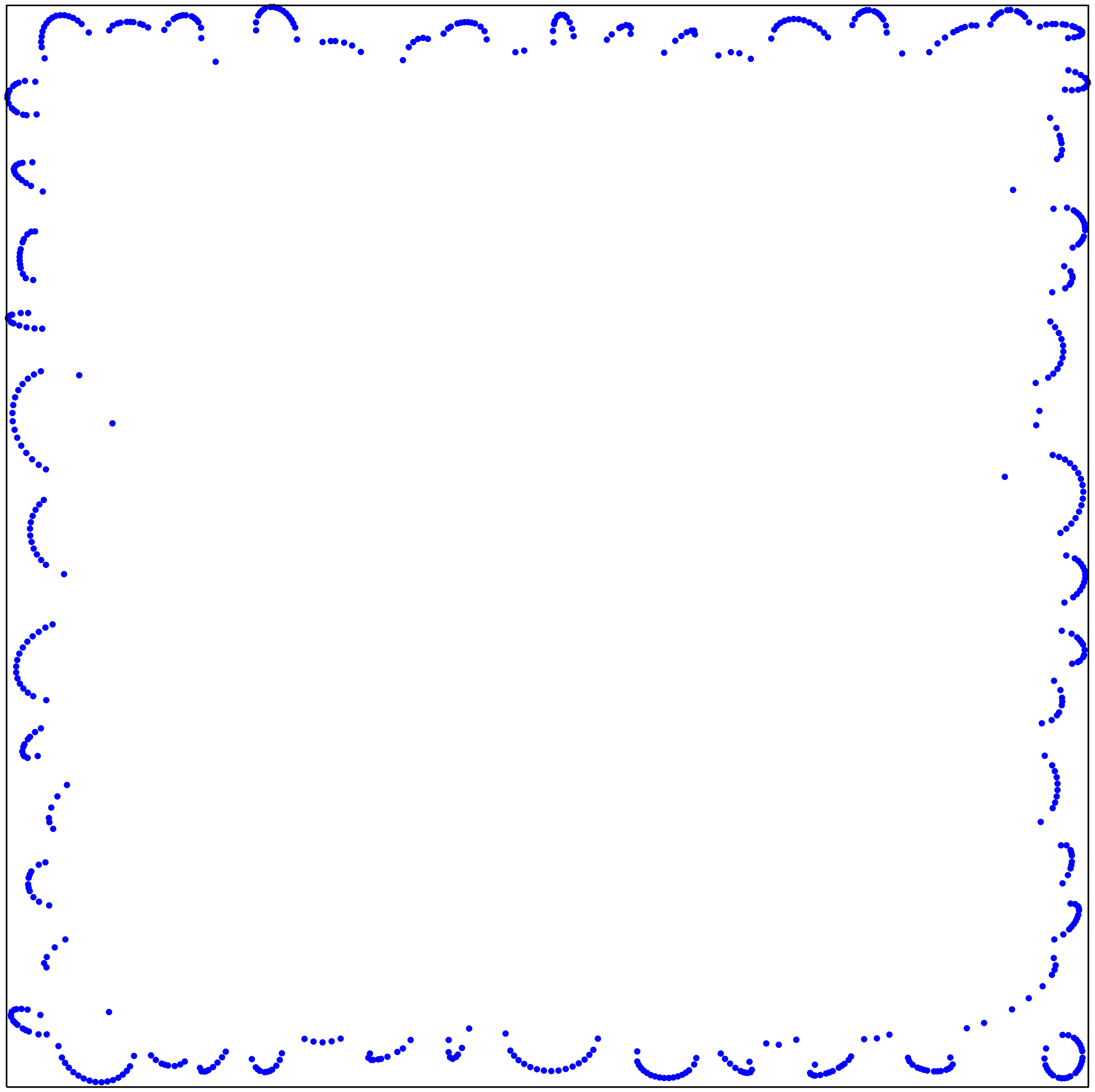
The micro-structure is modelled by a boundary integral equation defined on the internal boundaries. The internal degrees of freedom are eliminated hierarchically.

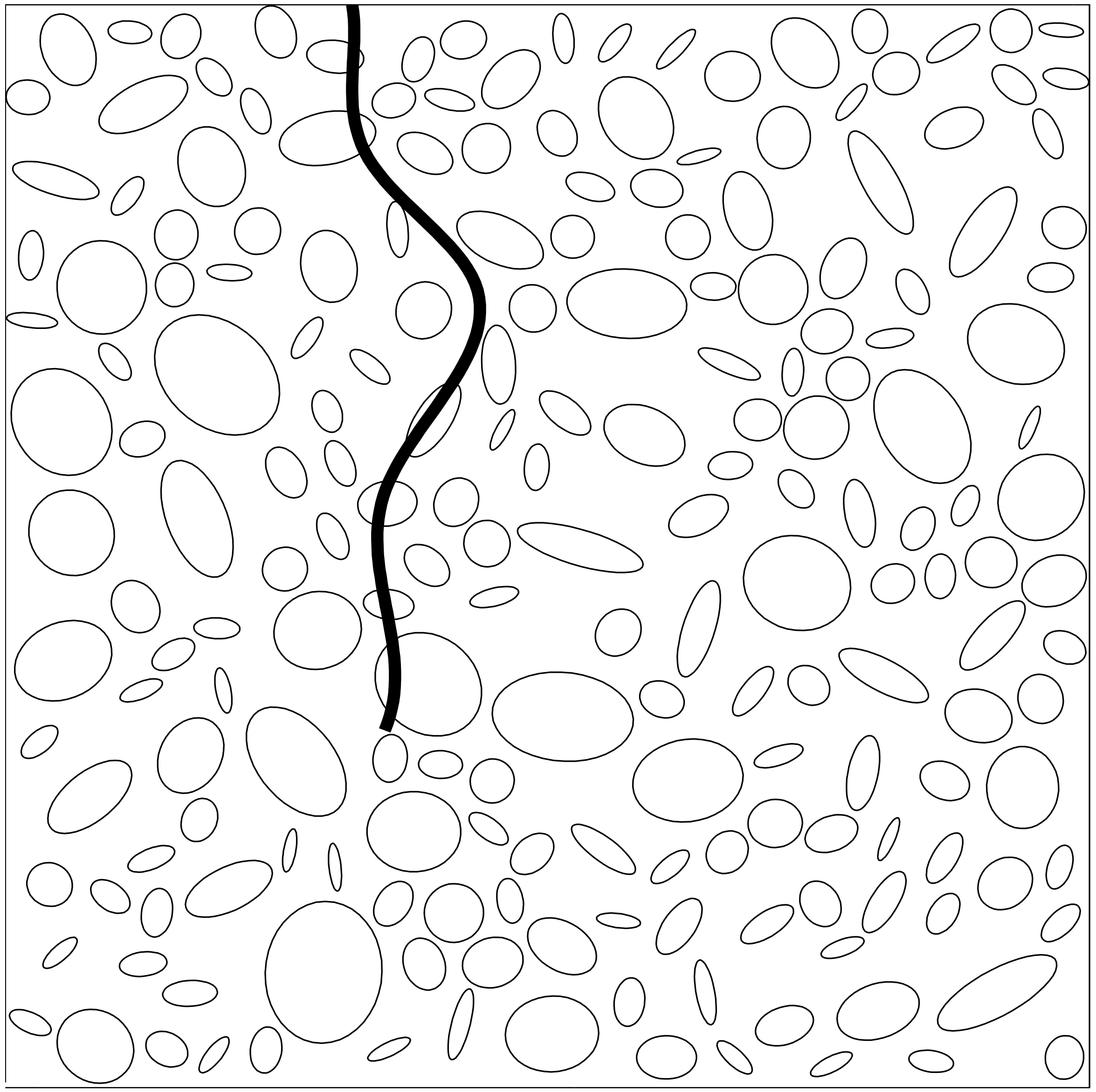




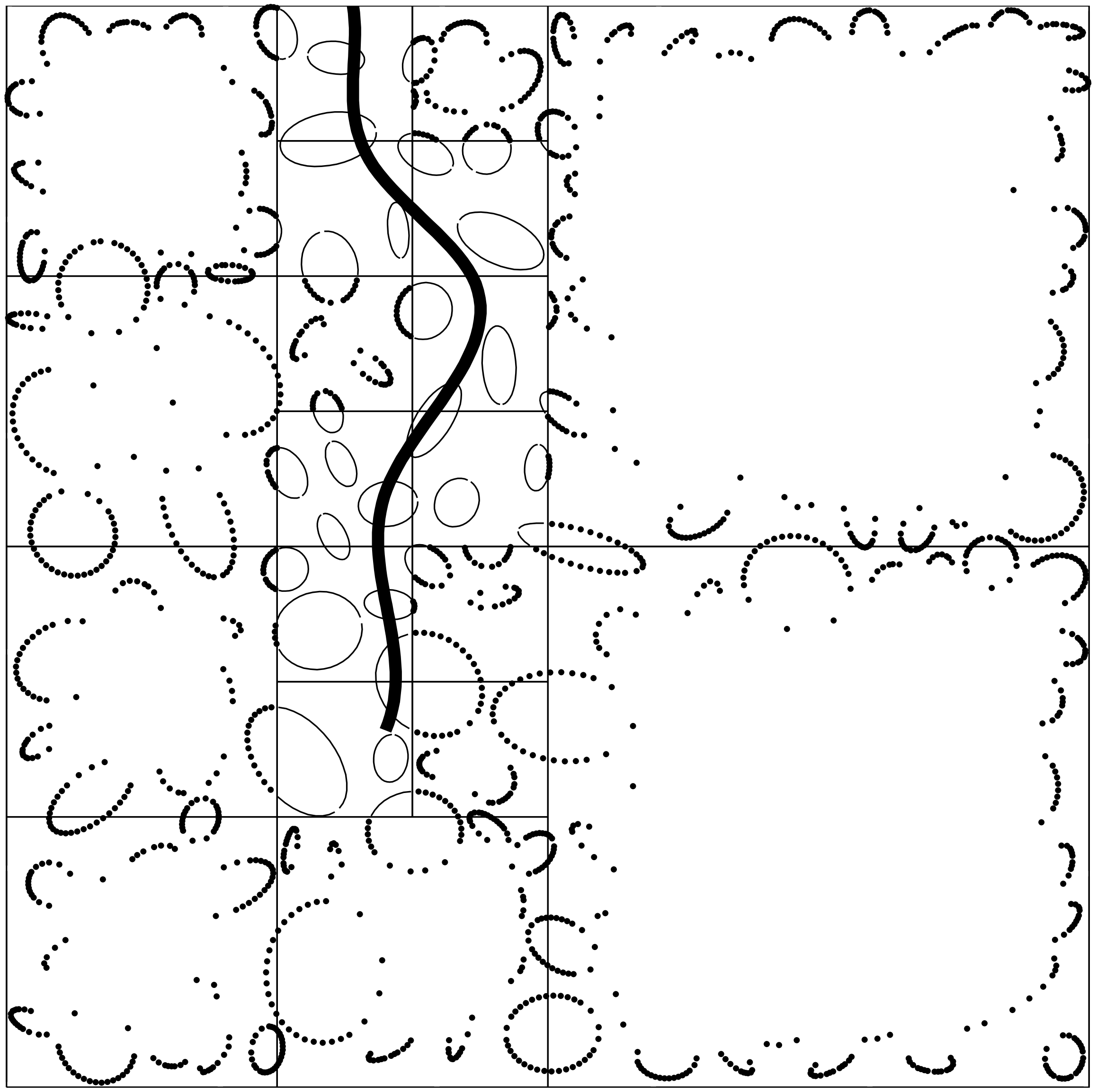






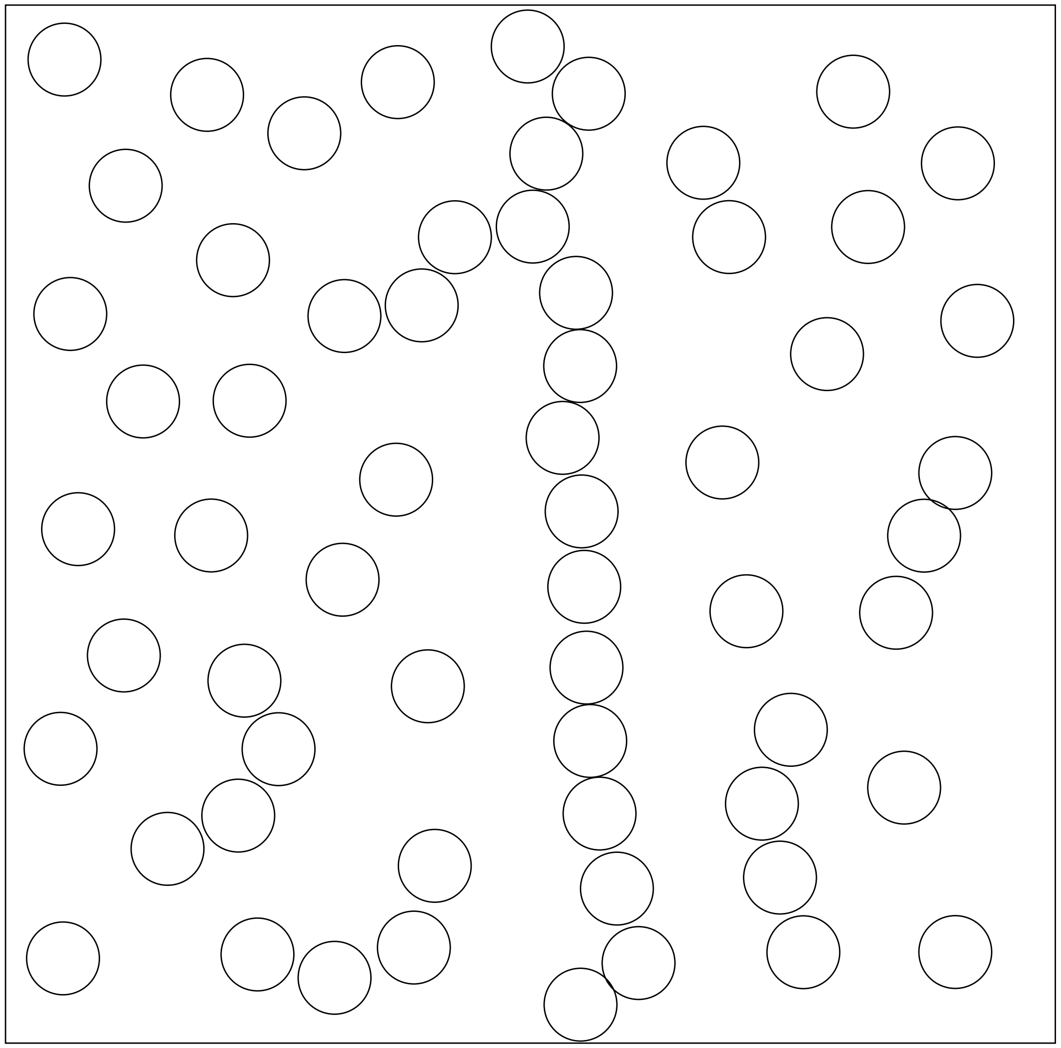


In the process, we computed the "Neumann-to-Dirichlet" operators for all the boxes in the quad tree. These can be very handy in computational modelling. Suppose we seek to model a crack propagating:



Any unaffected part of the material can be replaced by compressed operators.

The method can easily handle "percolating" micro-structures:



Concluding remarks:

- Efficient algorithms for computing approximations to the solution operator of elliptic linear PDEs have recently been developed.

- These techniques can with no further modifications be used to construct simplified models of problems involving complicated micro-structures.
  - Can handle complicated geometries and concentrated loads.
  - Particularly appealing in engineering design — the solution operators can be pre-computed and reused $\rightarrow$ *real time simulation.*
  - Solution operators can be merged, added, etc $\rightarrow$ *modular building blocks.*

- As of today, these techniques require the model to fully resolve the micro-structure. (Once!) They are in a sense *brute force* methods.

Future research directions:

- Construct fast algorithms for more environments. 3D still challenging.
- Construct the operators from statistical info. about the micro-structure.
- High-frequency problems . . .