Randomized algorithms and fast direct solvers

P.G. Martinsson, The University of Colorado at Boulder

Acknowledgements: Some of the work presented is joint work with Vladimir Rokhlin and Mark Tygert at Yale University.

In this talk, we will discuss numerical methods for solving the equation

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

where A is a linear constant-coefficient partial differential operator, and B is some local linear boundary operator ($e.g.$ Dirichlet B.C. $\Leftrightarrow B = I$).

Examples include:

- Laplace's equation.
- Helmholtz' equation.
- Stokes' equation.
- The Yukawa equation.
- The equations of linear elasticity.

Specifically, we will be concerned with the fast solution of the system of linear equations obtained upon discretization of (BVP).

There are two standard techniques for obtaining the discretized system:

Linear boundary value problem.

↙ ↘

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

↓

Direct discretization of the differential operator via Finite Element Method (FEM), Finite Difference Method, Finite Volume Method, . . .

Discretization of (BIE) using Nyström, collocation, Boundary Element Method, ...

↘ ↙

$N \times N$ system of linear algebraic equations.

1 – Methods based on discretizing PDEs:

Discretize the differential operator directly; instead of

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

solve

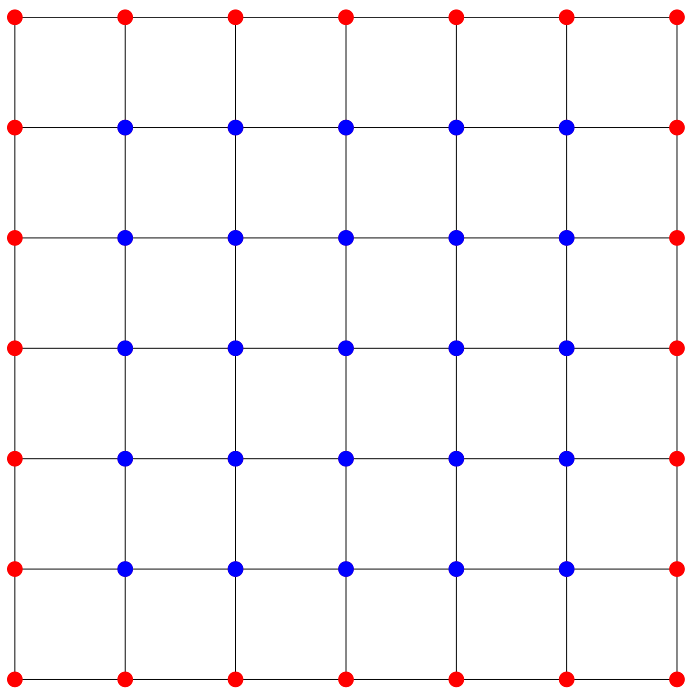
$$\text{(BVP-DISC)} \qquad A_N\,u_N = h_N,$$

where u_N is a function in an N-dimensional function space, A_N is an $N \times N$ matrix discretizing the operator A (obtained via Finite Elements / Finite Differences / ...), and h_N is a vector of data derived from f and g.

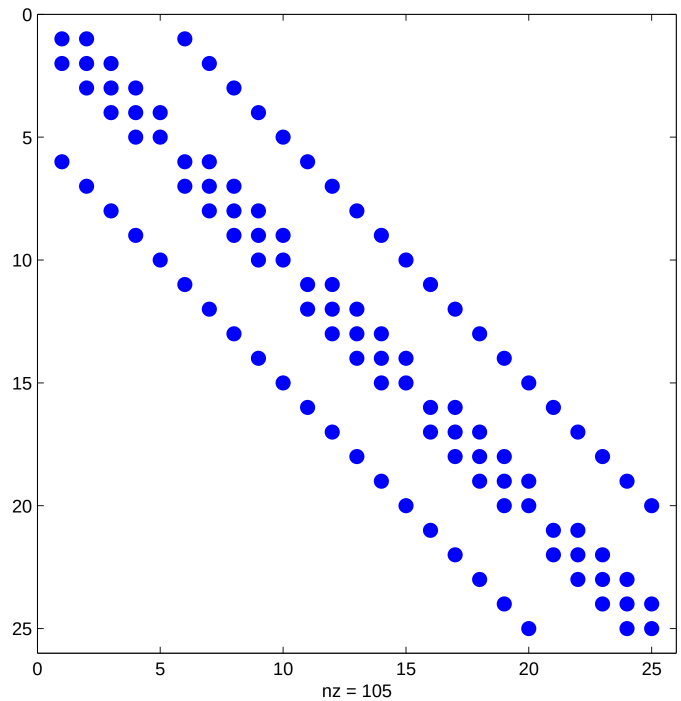
Example: Let Ω be a square, and let $A = -\Delta$. Discretize Ω into a grid, and let A_N denote the five-point stencil,

$$[A_N\varphi](i,j) = 4\varphi(i,j) - \varphi(i-1,j) - \varphi(i+1,j) - \varphi(i,j-1) - \varphi(i,j+1).$$

Then A_N is a sparse matrix.



The grid.



Sparsity pattern of A_N.

N is typically large, so solving

$$A_N\, u_N = h_N,$$

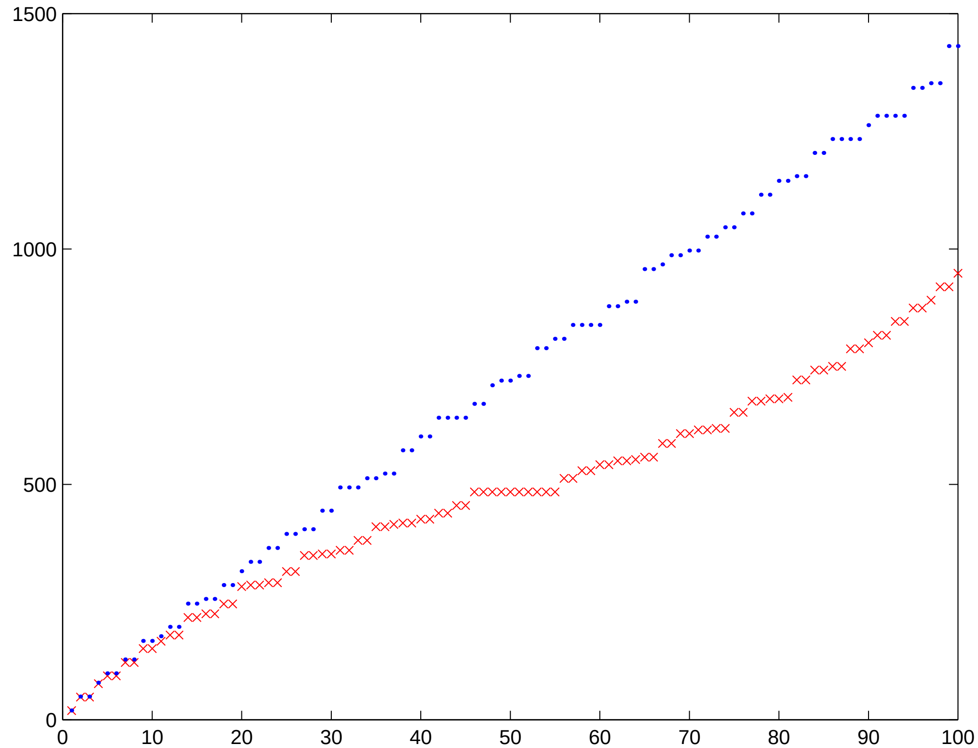
using Gaussian elimination would be very expensive. However, since A_N is sparse, we can use an iterative solver (conjugate gradients / GMRES / ...).

Problem:

A is an unbounded operator $\Rightarrow$

The matrix A_N is ill-conditioned $\Rightarrow$

The iterative solver converges slowly.



Pre-conditioners can help solving ill-conditioned linear systems.

A pre-conditioner is an operator P_N such that:

- It is cheap to apply P_N to a vector.
- The product $P_N\, A_N$ is well-conditioned.

Loosely speaking, $P_N \approx A_N^{-1}$.

The idea is to use an iterative solver to solve

$$P_N\, A_N\, u_N = P_N\, h_N.$$

The popular multigrid algorithm is a form of a pre-conditioner.

However, many problems related to ill-conditioning remain.

Would it be possible to directly compute A_N^{-1}?

2 – Methods based on discretizing integral equations:

Reformulate the BVP as a Boundary Integral Equation.

Example:

(BVP)
$$\begin{cases} -\Delta u(x) = 0, & x \in \Omega, \\ u(x) = f(x), & x \in \Gamma, \end{cases}$$

We make the following Ansatz:

$$u(x) = \int_\Gamma \big(n(y) \cdot \nabla_y \log|x-y|\big) v(y)\, ds(y), \qquad x \in \Omega,$$

where $n(y)$ is the outward pointing unit normal of Γ at y. Then the boundary charge distribution u satisfies the Boundary Integral Equation

(BIE)
$$v(x) + 2\int_\Gamma \big(n(y) \cdot \nabla_y \log|x-y|\big) v(y)\, ds(y) = 2f(x), \qquad x \in \Gamma.$$

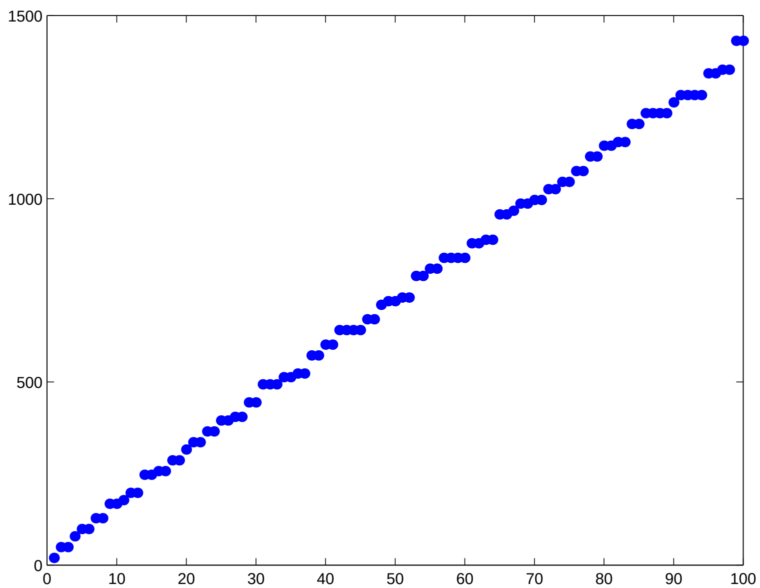
- (BIE) and (BVP) are in a strong sense equivalent.
- (BIE) is appealing mathematically (2^{nd} kind Fredholm equation).

Partial Differential Equation

$$-\Delta\, u = g$$

$-\Delta$ is an unbounded operator.

Typical spectrum of $-\Delta$:

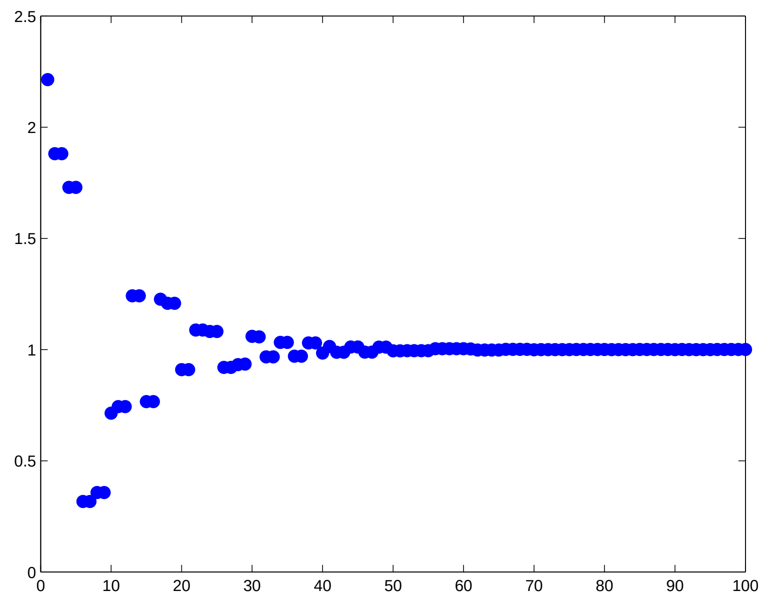


Second kind Fredholm Eqn.

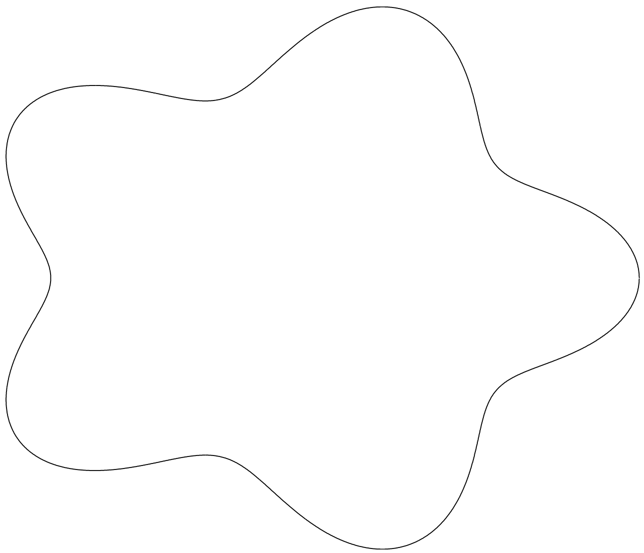
$$(I + K)\, u = f$$

K is a compact ("almost finite-dimensional") operator.

Typical spectrum of $I + K$:



The condition numbers of the *discretized* operators.



h	Cond. nr. of discretized BIE	
0.2	8.546837835256035	$(N = 25)$
0.1	7.053618952378199	$(N = 50)$
0.05	6.993154106860152	$(N = 100)$
0.025	6.993012937976997	$(N = 200)$
0.0125	6.993012936936591	$(N = 400)$
0.00625	6.993012936936595	$(N = 800)$

Rewriting the BVP as a BIE can be viewed as analytic pre-conditioning.

There is a fundamental problem with using integral operators in numerics:

> Discretization of integral operators typically results in dense matrices.

In the 1950's when computers made numerical PDE solvers possible, researchers faced a grim choice:

PDE-based:	Ill-conditioned, N is too large, low accuracy.
Integral Equations:	Dense system.

The integral equations lost and were largely forgotten

— they were simply too expensive.

(Except in some scattering problems where there was no choice.)

The situation changed dramatically in the 1980's. It was discovered that while K_N (the discretized integral operator) is dense, it is possible to evaluate the matrix-vector product

$$v \mapsto K_N\, v$$

in $O(N)$ operations – to high accuracy and with a small constant.

A very succesful such algorithm is the Fast Multipole Method by Rokhlin and Greengard (circa 1985).

A PRESCRIPTION FOR RAPIDLY SOLVING BVPS:

(BVP)
$$\begin{cases} -\Delta v(x) = 0, & x \in \Omega, \\ v(x) = f(x), & x \in \Gamma. \end{cases}$$

Convert (BVP) to a second kind Fredholm equation:

(BIE)
$$u(x) + \int_\Gamma \big(n(y) \cdot \nabla_y \log|x - y|\big) u(y)\, ds(y) = f(x), \qquad x \in \Gamma.$$

Discretize (BIE) into the discrete equation

(DISC)
$$(I + K_N) u_N = f_N$$

where K_N is a (typically dense) $N \times N$ matrix.

Fast Multipole Method — Can multiply K_N by a vector in $O(N)$ time.

Iterative solver — Solves (DISC) using $\sqrt{\kappa}$ matrix-vector multiplies, where κ is the condition number of $(I + K_N)$.

Total complexity — $O(\sqrt{\kappa}\, N)$. (Recall that κ is small. Like 14.)

We have described two paradigms for numerically solving BVPs:

$$\text{PDE formulation} \quad \Leftrightarrow \quad \text{Integral Equation formulation}$$

Which one should you choose?

When it is applicable, compelling arguments favor the use of the IE formulation:

Conditioning:
When there exists an IE formulation that is a Fredholm equation of the second kind, the mathematical equation itself is well-conditioned.

Dimensionality:
Frequently, an IE can be defined on the boundary of the domain.

Integral operators are benign objects:
It is (relatively) easy to implement high order discretizations of integral operators. Relative accuracy of 10^{-10} or better is often achieved.

However, integral equation based methods are quite often not a choice:

Fundamental limitations: They require the existence of a fundamental solution to the (dominant part of the) partial difference operator. In practise, this means that the (dominant part of the) operator must be linear and constant-coefficient.

Practical limitations: Underdeveloped infra-structure; there does not exist a general framework for discretizing surfaces. Lack of engineering strength codes. Etc.

To summarize:

- There exist $O(N)$ algorithms for a wide range of BVPs.

- For some BVPs, the N in $O(N)$ can be the number of degrees of freedom required to discretize the boundary.

- Regardless of how a BVP is discretized, it is not so simple to solve the resulting linear system.
  - Finite Element Methods: system is sparse but ill-conditioned.
  - Boundary Integral Methods: system is dense.

- Almost all existing $O(N)$ methods rely on *iterative* solvers.

In some environments, the linear solve presents a serious challenge:

1. Problems whose geometry require a very large number of unknowns:
   - Modeling of heterogeneous materials.
   - Radar scattering off of the ocean surface.

2. Applications that require a very large number of solves:
   - Molecular Dynamics.
   - Optimal design.

3. Problems that are inherently ill-conditioned:
   - Scattering problems at intermediate or high frequencies.
   - Ill-conditioning due to geometry (elongated domains, percolation, etc).

The inadequacy of existing methods in these environments stems from their reliance on iterative solvers. We need **direct** solvers.

What is a direct solver?

Recall that many BVPs can be cast in the following form:

$$\text{(BIE)} \qquad u(x) + \int_\Gamma g(x,y)u(y)\,ds(y) = f(x), \qquad x \in \Gamma.$$

Upon discretization, equation (BIE) turns into a discrete equation

$$\text{(DISC)} \qquad (I + K_N)u = f$$

where K_N is a (typically dense) $N \times N$ matrix.

A ***direct method*** computes a compressed representation for $(I + K_N)^{-1}$.

- Cost for pre-computing the inverse.
- Cost for applying the inverse to a vector.

In many environments, both of these costs can be made $O(N)$.

Direct methods are good for (1) ill-conditioned problems, (2) problems with multiple right-hand sides, (3) spectral decompositions, (4) updating, ...

Practical 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.

Sampling of related work:

1991 Sparse matrix algebra / wavelets, *Beylkin, Coifman, Rokhlin*,

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.*

Current state of the research

The fast direct solvers we are developing exploit the fact that off-diagonal blocks of the matrix to be inverted have low rank.

This restricts the range of application to non-oscillatory, or moderately oscillatory problems. In other words, we **can** do:

- Laplace's equation, equations of elasticity, Yukawa's equation,. . .
- Helmholtz' and Maxwell's equations for low and intermediate frequencies. (In special cases, high frequency problem can also be solved.)

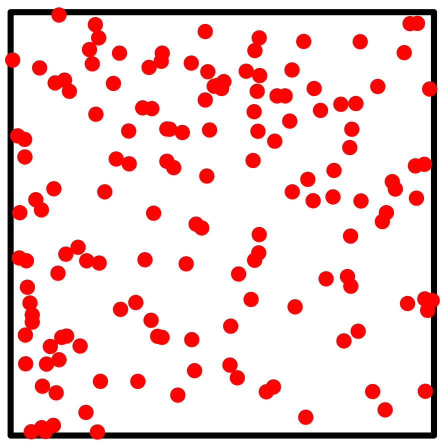
Linear time algorithms currently exist for:

- Boundary integral equations in 2D (published).
- Finite element / finite difference matrices in 2D (code exists).
- Lippmann-Schwinger equations in 2D (code under way).

In 3D, we "know" how to solve the problem, but much work remains.

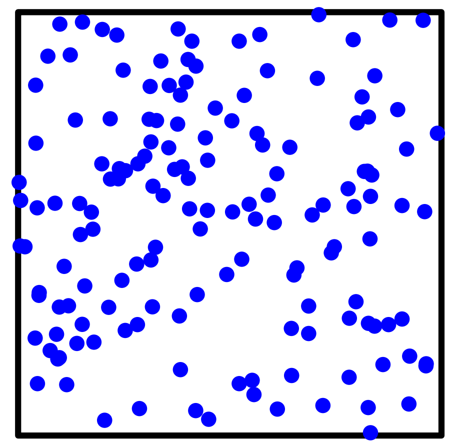
How does it work? Like classical FMM, it is based on rank-deficiencies

— *large blocks of the dense matrix have low-rank approximants.*



N source points y_n in Ω_2.

Given charges $\{q_n\}_{n=1}^N$.



M target points x_m in Ω_1.

Sought potentials $\{v_m\}_{m=1}^M$.

We want to evaluate (in complex arithmetic):

$$v_m = \sum_{n=1}^{N} \log(x_m - y_n)\, q_n, \qquad \text{for } m = 1, \dots, M.$$

Let A_{12} denote the $M \times N$ matrix with entries $\log(x_m - y_n)$.

$$\{q_n\}_{n=1}^N \xrightarrow{A_{12}} \{v_m\}_{m=1}^M$$

The matrix A_{12} is *dense*, so the cost is $O(M\,N)$.

Multipole expansion: To precision ε, we wish to evaluate

$$v_m = \sum_{n=1}^{N} \log(x_m - y_n)\, q_n, \qquad \text{for } m = 1, \dots, M. \tag{11}$$

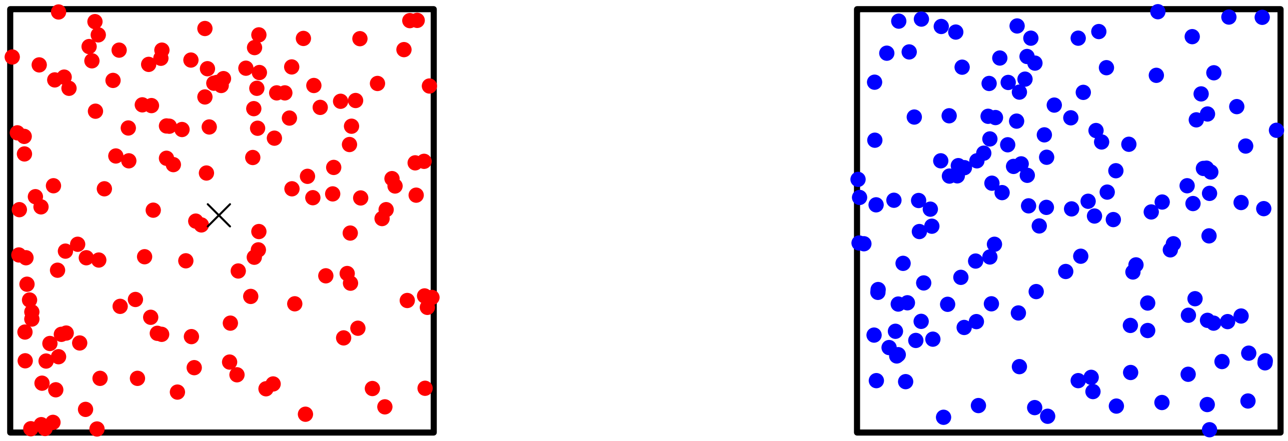
If $|x_m| \geq R_1$ and $|y_n| \leq R_2$, then

$$\log(x_m - y_n) = \log(x_m) - \sum_{j=1}^{k} \frac{y_n^j}{n\, x_m^j} + O\left(\left(\frac{R_2}{R_1}\right)^{k+1}\right). \tag{12}$$

Combining (11) and (12) we obtain

$$v_m = \underbrace{\left(\sum_{n=1}^{N} q_n\right)}_{=\alpha_0} \log(x_m) + \sum_{j=1}^{k} \underbrace{\left(\sum_{n=1}^{N} -\frac{y_n^j}{n}\right)}_{=\alpha_j} \frac{1}{x_m^j} + \varepsilon.$$

- Step 1: Compute $\{\alpha_j\}_{j=1}^{k}$ from $\{q_n\}_{n=1}^{N}$. Cost $\sim k\, N$.
- Step 2: Compute $\{v_m\}_{m=1}^{M}$ from $\{\alpha_j\}_{j=1}^{k}$. Cost $\sim k\, M$.

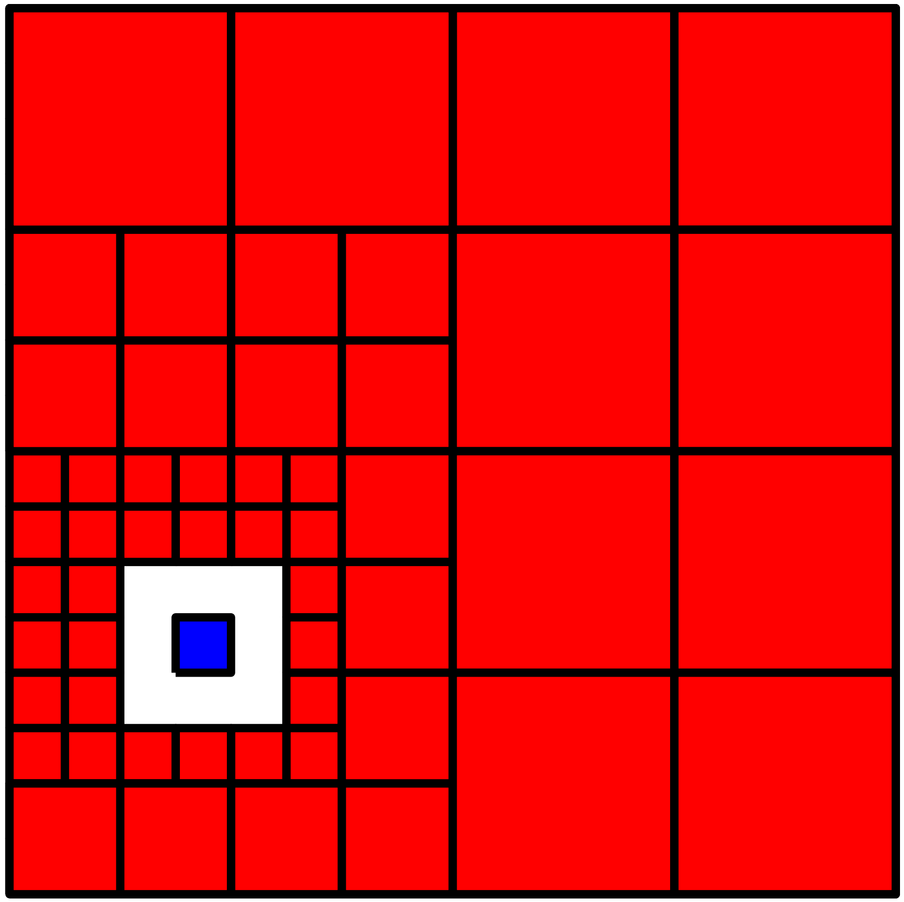


$$\begin{tikzcd}
\{q_n\}_{n=1}^{N} \arrow[r, "A_{12}"] \arrow[d] & \{v_m\}_{m=1}^{M} \\
\{\alpha_j\}_{j=1}^{k} \arrow[ur] &
\end{tikzcd}$$

We have reduced the computational cost from $O(M\,N)$ to $O(k\,(M+N))$.

The requested accuracy ε satisfies $\varepsilon \sim \left(\frac{R/\sqrt{2}}{3R/2}\right)^{k+1}$, so $k \sim \frac{\log|\varepsilon|}{\log(3/\sqrt{2})}$.

When the target region and the source regions are the same, we tessellate space into a hierarchy of boxes.



A naïve implementation (the "Barnes-Hut" scheme) leads to: Cost $\sim |\log \varepsilon|\,(\log\, N)\, N$

With some refinements (the "Fast Multipole Method") we obtain: Cost $\sim |\log \varepsilon|\, N$.

Let us return to the direct solvers.

Consider the linear system

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}.$$

Suppose that for $i \neq j$, the blocks A_{ij} allow the factorization

$$\underbrace{A_{ij}}_{n_i \times n_j} = \underbrace{U_i}_{n_i \times k_i} \, \underbrace{\tilde{A}_{ij}}_{k_i \times k_j} \, \underbrace{U_j^{\mathrm{t}}}_{k_j \times n_j} .$$

The ranks k_i are significantly smaller than the block sizes n_i.

We then let

$$\underbrace{\tilde{q}_i}_{k_i \times 1} = U_i^{\mathrm{t}} \, \underbrace{q_i}_{n_i \times 1},$$

be the variables of the "reduced" system.

Recall: $A_{ij} = U_i\,\tilde{A}_{ij}\,U_j^{\mathrm{t}}$ and $\tilde{q}_i = U_i^{\mathrm{t}}\,q_i$.

The system $\sum_j A_{ij}\,q_j = v_i$ then takes the form

$$\left[\begin{array}{cccc|cccc}
A_{11} & 0 & 0 & 0 & 0 & U_1\tilde{A}_{12} & U_1\tilde{A}_{13} & U_1\tilde{A}_{14} \\
0 & A_{22} & 0 & 0 & U_2\tilde{A}_{21} & 0 & U_2\tilde{A}_{23} & U_2\tilde{A}_{24} \\
0 & 0 & A_{33} & 0 & U_3\tilde{A}_{31} & U_3\tilde{A}_{32} & 0 & U_3\tilde{A}_{34} \\
0 & 0 & 0 & A_{44} & U_4\tilde{A}_{41} & U_4\tilde{A}_{42} & U_4\tilde{A}_{43} & 0 \\
\hline
-U_1^{\mathrm{t}} & 0 & 0 & 0 & I & 0 & 0 & 0 \\
0 & -U_2^{\mathrm{t}} & 0 & 0 & 0 & I & 0 & 0 \\
0 & 0 & -U_3^{\mathrm{t}} & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & -U_4^{\mathrm{t}} & 0 & 0 & 0 & I
\end{array}\right]
\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ \tilde{q}_1 \\ \tilde{q}_2 \\ \tilde{q}_3 \\ \tilde{q}_4 \end{bmatrix}
=
\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Now form the Schur complement to eliminate the q_j's.

After eliminating the "fine-scale" variables q_i, we obtain

$$\left[\begin{array}{cccc} I & U_1^{\rm t}\tilde{A}_{11}^{-1}U_1\tilde{A}_{12} & U_1^{\rm t}\tilde{A}_{11}^{-1}U_1\tilde{A}_{13} & U_1^{\rm t}\tilde{A}_{11}^{-1}U_1\tilde{A}_{14} \\ U_2^{\rm t}\tilde{A}_{22}^{-1}U_2\tilde{A}_{21} & I & U_2^{\rm t}\tilde{A}_{22}^{-1}U_2\tilde{A}_{23} & U_2^{\rm t}\tilde{A}_{22}^{-1}U_2\tilde{A}_{24} \\ U_3^{\rm t}\tilde{A}_{33}^{-1}U_3\tilde{A}_{31} & U_3^{\rm t}\tilde{A}_{33}^{-1}U_3\tilde{A}_{32} & I & U_3^{\rm t}\tilde{A}_{33}^{-1}U_3\tilde{A}_{34} \\ U_4^{\rm t}\tilde{A}_{44}^{-1}U_4\tilde{A}_{41} & U_4^{\rm t}\tilde{A}_{44}^{-1}U_4\tilde{A}_{42} & U_4^{\rm t}\tilde{A}_{44}^{-1}U_4\tilde{A}_{43} & I \end{array}\right] \left[\begin{array}{c} \tilde{q}_1 \\ \tilde{q}_2 \\ \tilde{q}_3 \\ \tilde{q}_4 \end{array}\right] = \left[\begin{array}{c} U_1^{\rm t}A_{11}^{-1}\,v_1 \\ U_2^{\rm t}A_{22}^{-1}\,v_2 \\ U_3^{\rm t}A_{33}^{-1}\,v_3 \\ U_4^{\rm t}A_{44}^{-1}\,v_4. \end{array}\right]$$

We set

$$\tilde{A}_{ii} = \left(U_i^{\rm t}\,A_{ii}^{-1}\,U_i\right)^{-1},$$

and multiply line i by $\tilde{A}_{ii}$ to obtain the reduced system

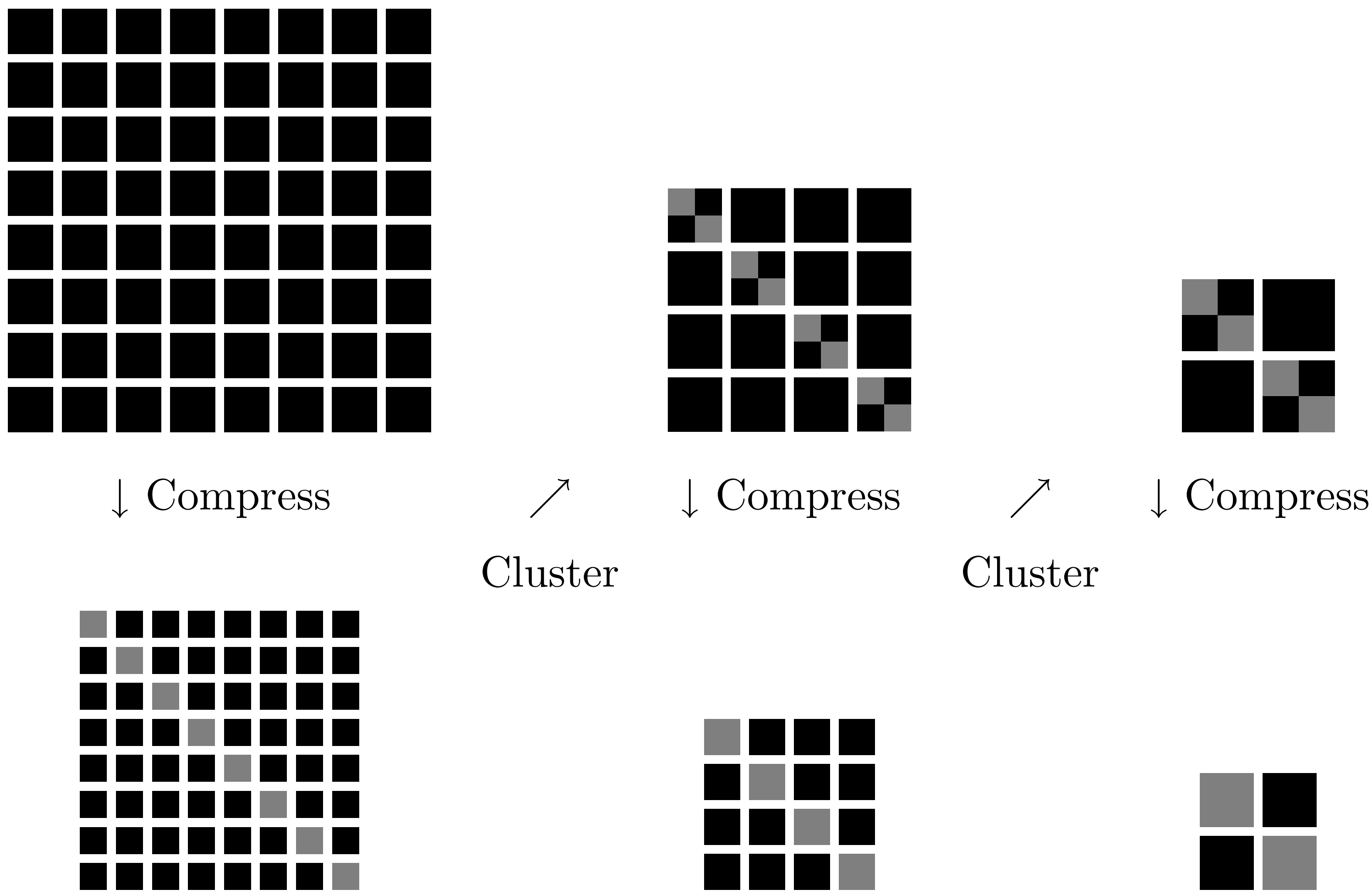
$$\left[\begin{array}{cccc} \tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} & \tilde{A}_{14} \\ \tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} & \tilde{A}_{24} \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} & \tilde{A}_{34} \\ \tilde{A}_{41} & \tilde{A}_{42} & \tilde{A}_{43} & \tilde{A}_{44} \end{array}\right] \left[\begin{array}{c} \tilde{q}_1 \\ \tilde{q}_2 \\ \tilde{q}_3 \\ \tilde{q}_4 \end{array}\right] = \left[\begin{array}{c} \tilde{v}_1 \\ \tilde{v}_2 \\ \tilde{v}_3 \\ \tilde{v}_4 \end{array}\right].$$

where

$$\tilde{v}_i = \tilde{A}_{ii}\,U_i^{\rm t}\,A_{ii}^{-1}\,v_i.$$

(This derivation was pointed out by Leslie Greengard.)

A globally $O(N)$ algorithm is obtained by hierarchically repeating the process:



The one-level coarse-graining involves the following steps:

- Compute U_i and $\tilde{A}_{ij}$ so that $A_{ij} = U_i\,\tilde{A}_{ij}\,U_j^{\mathrm{t}}$.

- Compute the new diagonal matrices

$$\tilde{A}_{ii} = \left(U_i^{\mathrm{t}}\,A_{ii}^{-1}\,U_i\right)^{-1},$$

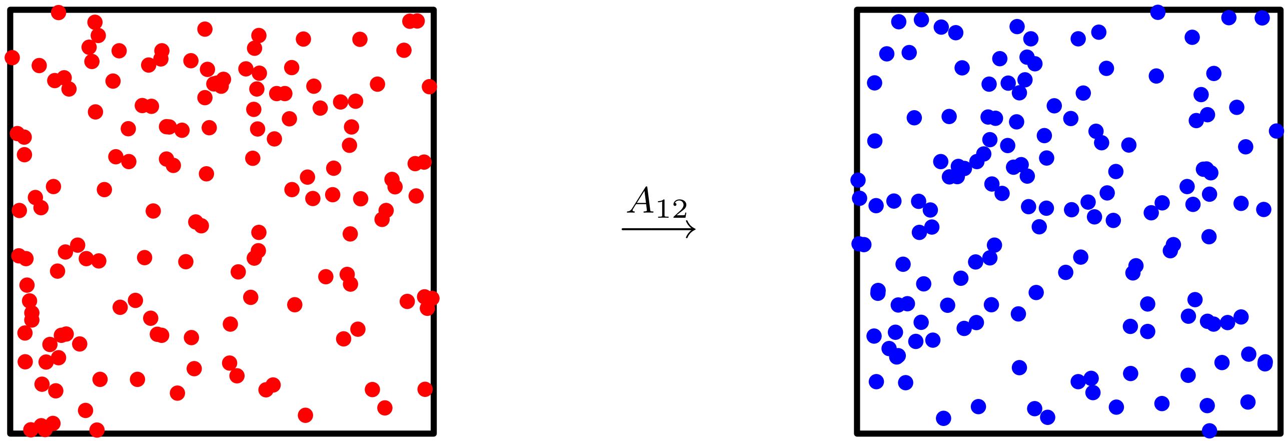
- Compute the new loads

$$\tilde{v}_i = \tilde{A}_{ii}\,U_i^{\mathrm{t}}\,A_{ii}^{-1}\,v_i.$$

For the algorithm to be $O(N)$, it has to be able to carry out these steps *locally*.

To achieve this, we use interpolative representations.

$\boxed{\tilde{A}_{ij} \text{ will be a submatrix of } A_{ij}\text{, so it will not need to be computed.}}$

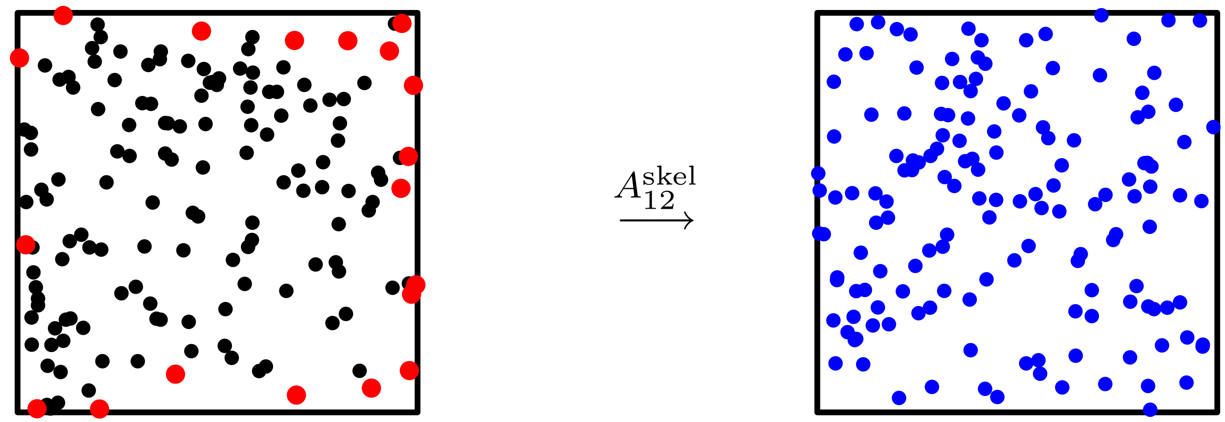


Sources $\{q_n\}_{n=1}^{N}$ Potentials $\{v_m\}_{m=1}^{M}$

$$\begin{tikzcd}
\{q_n\}_{n=1}^{N} \arrow[rr, "A_{12}"] \arrow[d, "?"'] & & \{v_m\}_{m=1}^{M} \\
\text{``Small'' representation} \arrow[urr, "?"'] & &
\end{tikzcd}$$

The key observation is that $k = \mathrm{rank}(A_{12}) < \min(M, N)$.

Skeletonization

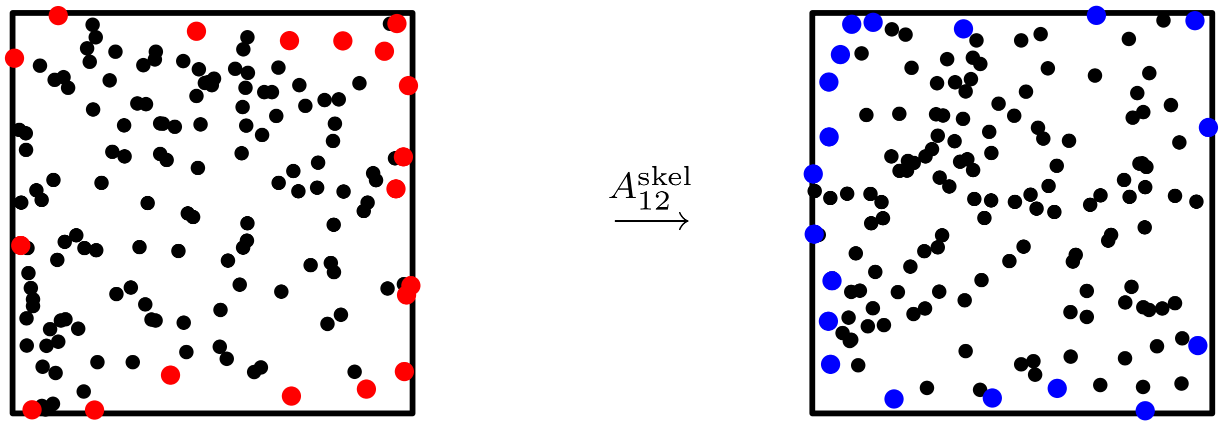


$$
\begin{tikzcd}
\{q_n\}_{n=1}^{N} \arrow[r, "A_{12}"] \arrow[d, "U_2^{\mathrm{t}}"'] & \{v_m\}_{m=1}^{M} \\
\{\tilde{q}_{n_j}\}_{j=1}^{k} \arrow[ur, "A_{12}^{\mathrm{skel}}"'] &
\end{tikzcd}
$$

We can pick k points in Ω_{S} with the property that any potential in Ω_{T} can be replicated by placing charges on these k points.

- The choice of points does not depend on $\{q_n\}_{n=1}^{N}$.
- A_{12}^{skel} is a submatrix of A_{12}.

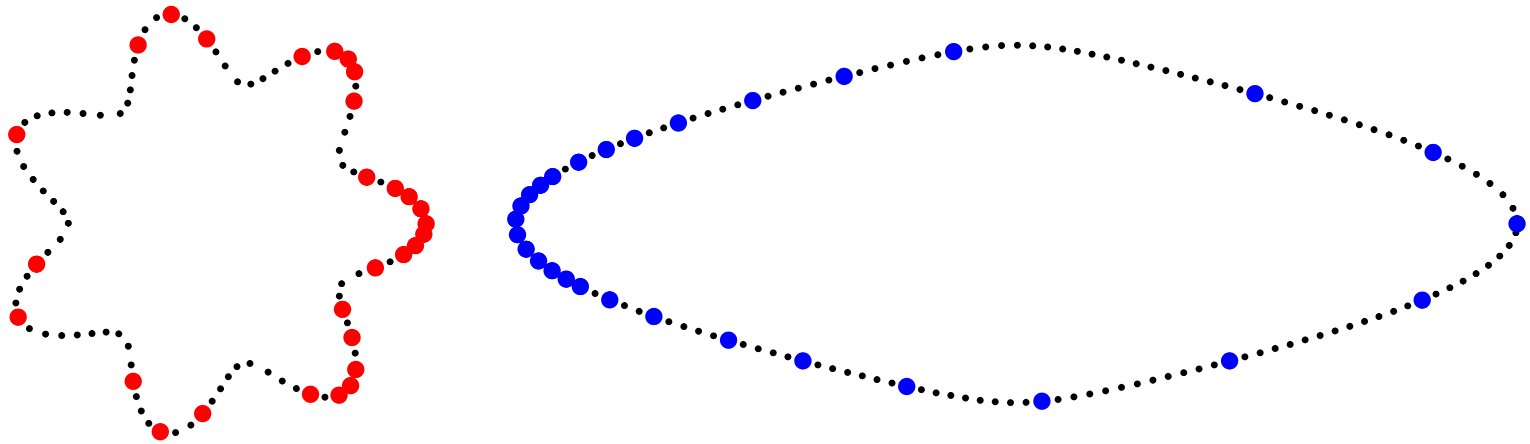
We can "skeletonize" both Ω_1 and Ω_2.



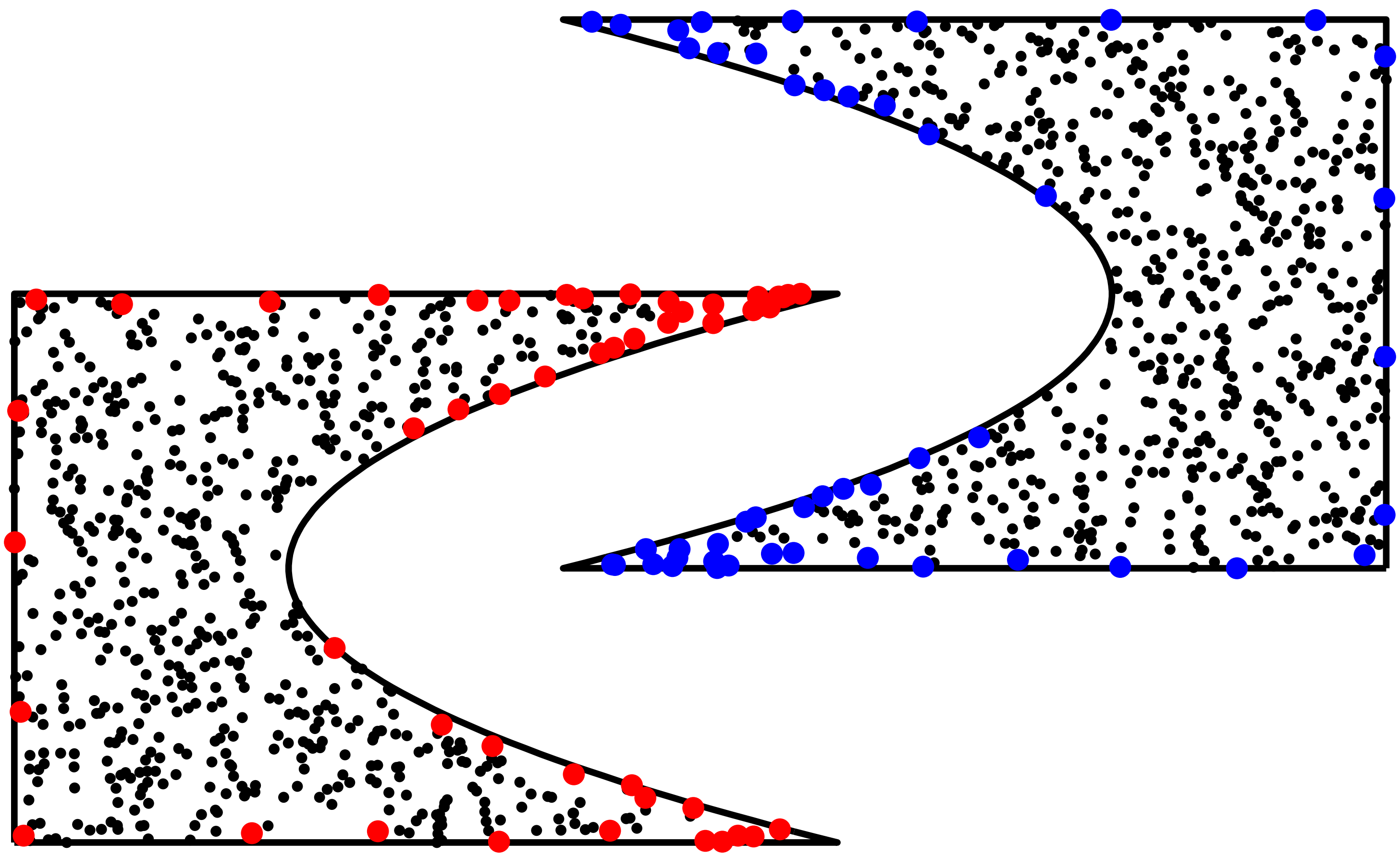
$$
\begin{tikzcd}
\{q_n\}_{n=1}^{N} \arrow[r, "A_{12}"] \arrow[d, "U_2^{\mathrm{t}}"'] & \{v_m\}_{m=1}^{M} \\
\{\tilde{q}_{n_j}\}_{j=1}^{k} \arrow[r, "A_{12}^{\text{skel}}"'] & \{v_{m_j}\}_{j=1}^{k} \arrow[u, "U_1"']
\end{tikzcd}
$$

Rank $= 19$ at $\varepsilon = 10^{-10}$.

Skeletonization can be performed for Ω_{S} and Ω_{T} of various shapes.

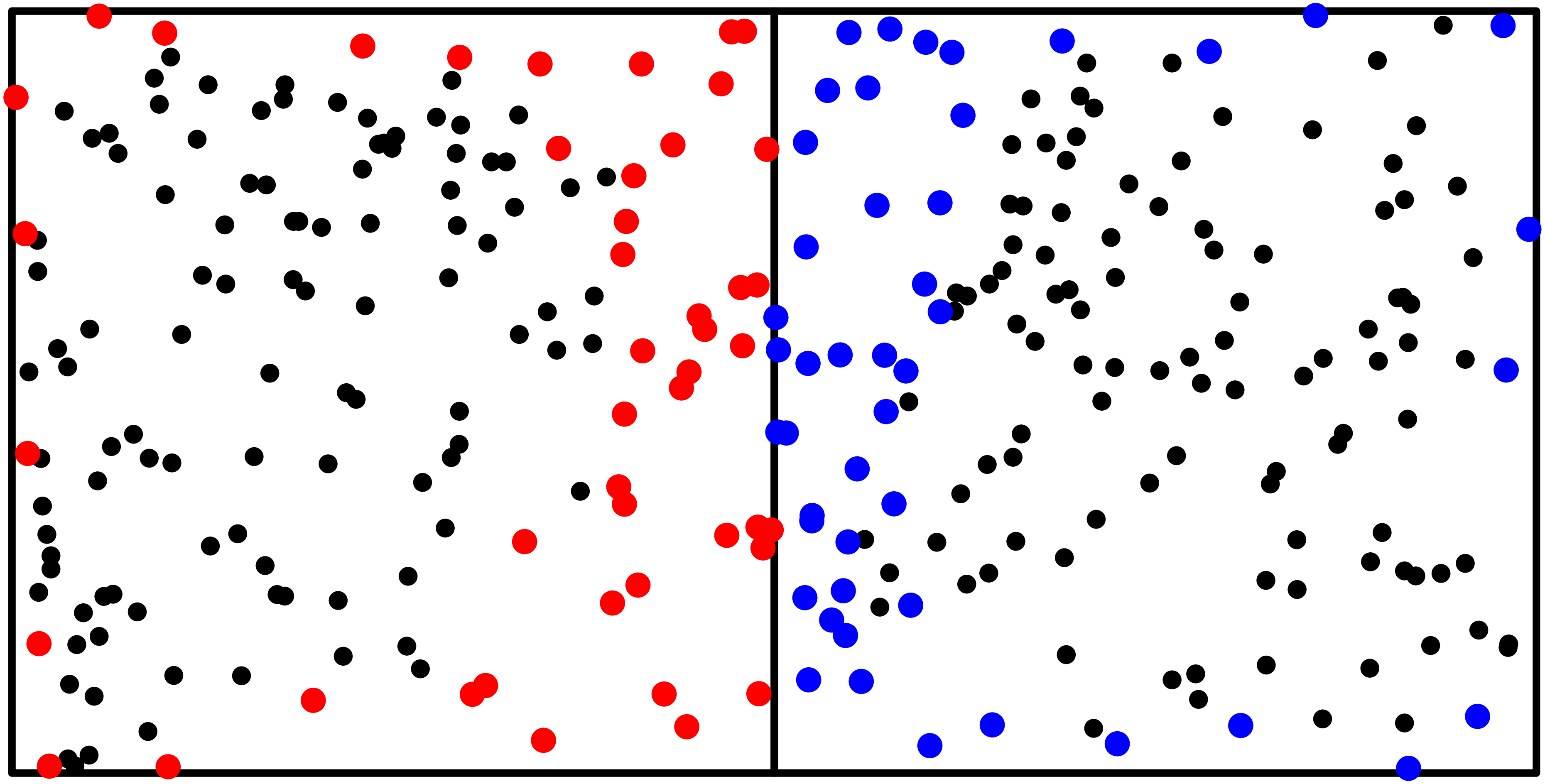


Rank $= 29$ at $\varepsilon = 10^{-10}$.



Rank $= 48$ at $\varepsilon = 10^{-10}$.

Adjacent boxes can be skeletonized.



Rank $= 46$ at $\varepsilon = 10^{-10}$.

$$
\begin{tikzcd}
\{q_n\}_{n=1}^{N} \arrow[r, "A_{12}"] \arrow[d, "U_2^{\mathrm{t}}"'] & \{v_m\}_{m=1}^{M} \\
\{\tilde{q}_{n_j}\}_{j=1}^{k} \arrow[r, "A_{12}^{\mathrm{skel}}"'] & \{v_{m_j}\}_{j=1}^{k} \arrow[u, "U_1"']
\end{tikzcd}
$$

Benefits:

- The rank is optimal.
- The projection and interpolation are cheap.
  U_1 and U_2 contain $k \times k$ identity matrices.
- The projection and interpolation are well-conditioned.
- Finding the k points is cheap.
- The map $\tilde{A}_{12}$ is simply a restriction of the original map A_{12}.
  (We can loosely say that "the physics of the problem is preserved".)
- Interaction between **adjacent** boxes can be compressed
  (no buffering is required).

Similar schemes have been proposed by many researchers:

1993 - C.R. Anderson

1995 - C.L. Berman

1996 - E. Michielssen, A. Boag

1999 - J. Makino

2004 - L. Ying, G. Biros, D. Zorin

A mathematical foundation:

1996 - M. Gu, S. Eisenstat

Let us return to the direct solver environment. Recall:

We convert the system

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

to the reduced system

$$\begin{bmatrix} \tilde{A}_{11} & A_{12}^{\text{skel}} & A_{13}^{\text{skel}} & A_{14}^{\text{skel}} \\ A_{21}^{\text{skel}} & \tilde{A}_{22} & A_{23}^{\text{skel}} & A_{24}^{\text{skel}} \\ A_{31}^{\text{skel}} & A_{32}^{\text{skel}} & \tilde{A}_{33} & A_{34}^{\text{skel}} \\ A_{41}^{\text{skel}} & A_{42}^{\text{skel}} & A_{43}^{\text{skel}} & \tilde{A}_{44} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \end{bmatrix} = \begin{bmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \tilde{f}_3 \\ \tilde{f}_4 \end{bmatrix}.$$

We know that A_{ij}^{skel} is a submatrix of A_{ij} when $i \neq j$.

What is $\tilde{A}_{ii}$?

We recall that the new diagonal blocks are defined by

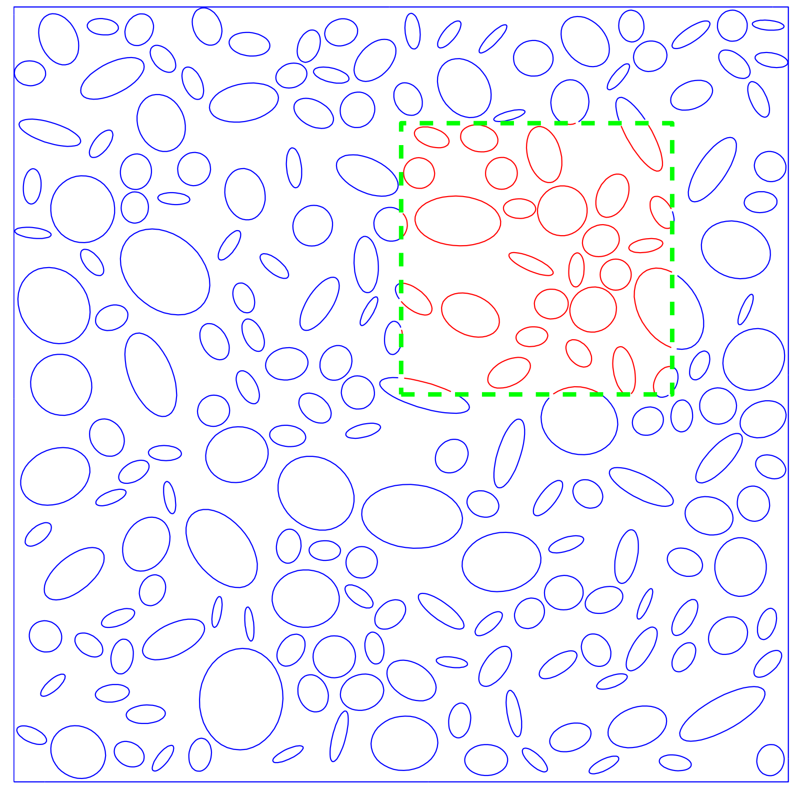
$$\underbrace{\tilde{A}_{ii}}_{k\times k} = \Big(\underbrace{U_i^{\rm t}}_{k\times n}\,\underbrace{A_{ii}^{-1}}_{n\times n}\,\underbrace{U_i}_{n\times k}\Big)^{-1}.$$

We call these blocks "proxy matrices".

What are they?

Let Ω_1 denote the block marked in red.

Let Ω_2 denote the rest of the domain.

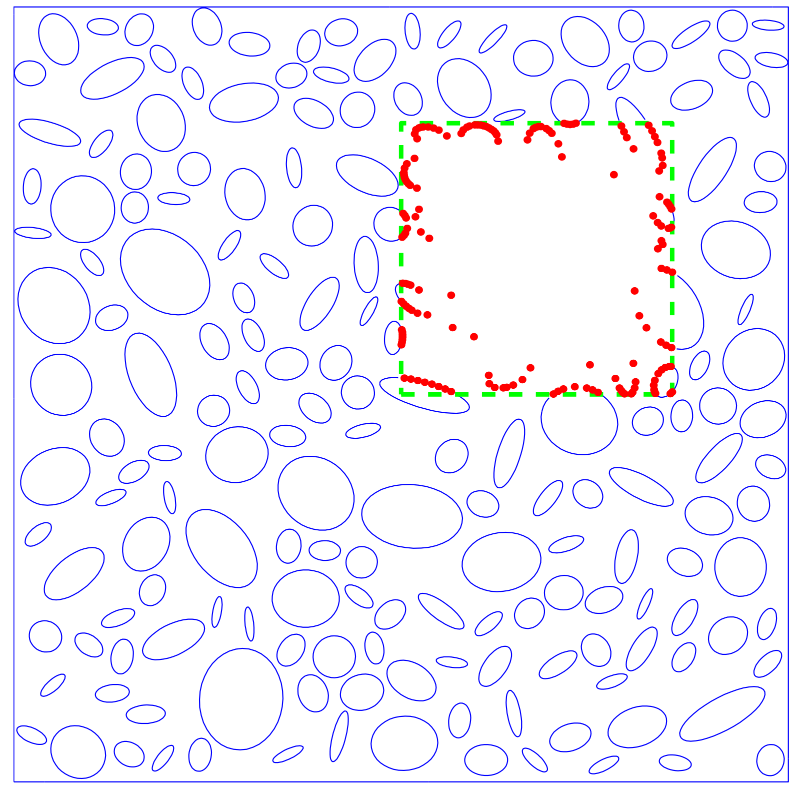


$$\begin{tikzcd}
\text{Charges on } \Omega_2 \arrow[r, "A_{12}"] \arrow[dr, "A_{12}^{\rm skel}"'] & \text{Pot. on } \Omega_1 \arrow[r, "A_{11}^{-1}"] & \text{Charges on } \Omega_1 \arrow[r, "A_{21}"] \arrow[d, "U_1^{\rm t}"'] & \text{Pot. on } \Omega_2 \\
 & \text{Pot. on } \Omega_1^{\rm skel} \arrow[u, "U_1"'] \arrow[r, "\tilde{A}_{11}^{-1}"] & \text{Charges on } \Omega_1^{\rm skel} \arrow[ur, "A_{12}^{\rm skel}"'] &
\end{tikzcd}$$

$\tilde{A}_{11}$ contains *all the information the outside world needs to know about* Ω_1.

We recall that the new diagonal blocks are defined by

$$\underbrace{\tilde{A}_{ii}}_{k\times k} = \Big(\underbrace{U_i^{\rm t}}_{k\times n}\,\underbrace{A_{ii}^{-1}}_{n\times n}\,\underbrace{U_i}_{n\times k}\Big)^{-1}.$$



We call these blocks "proxy matrices".

What are they?

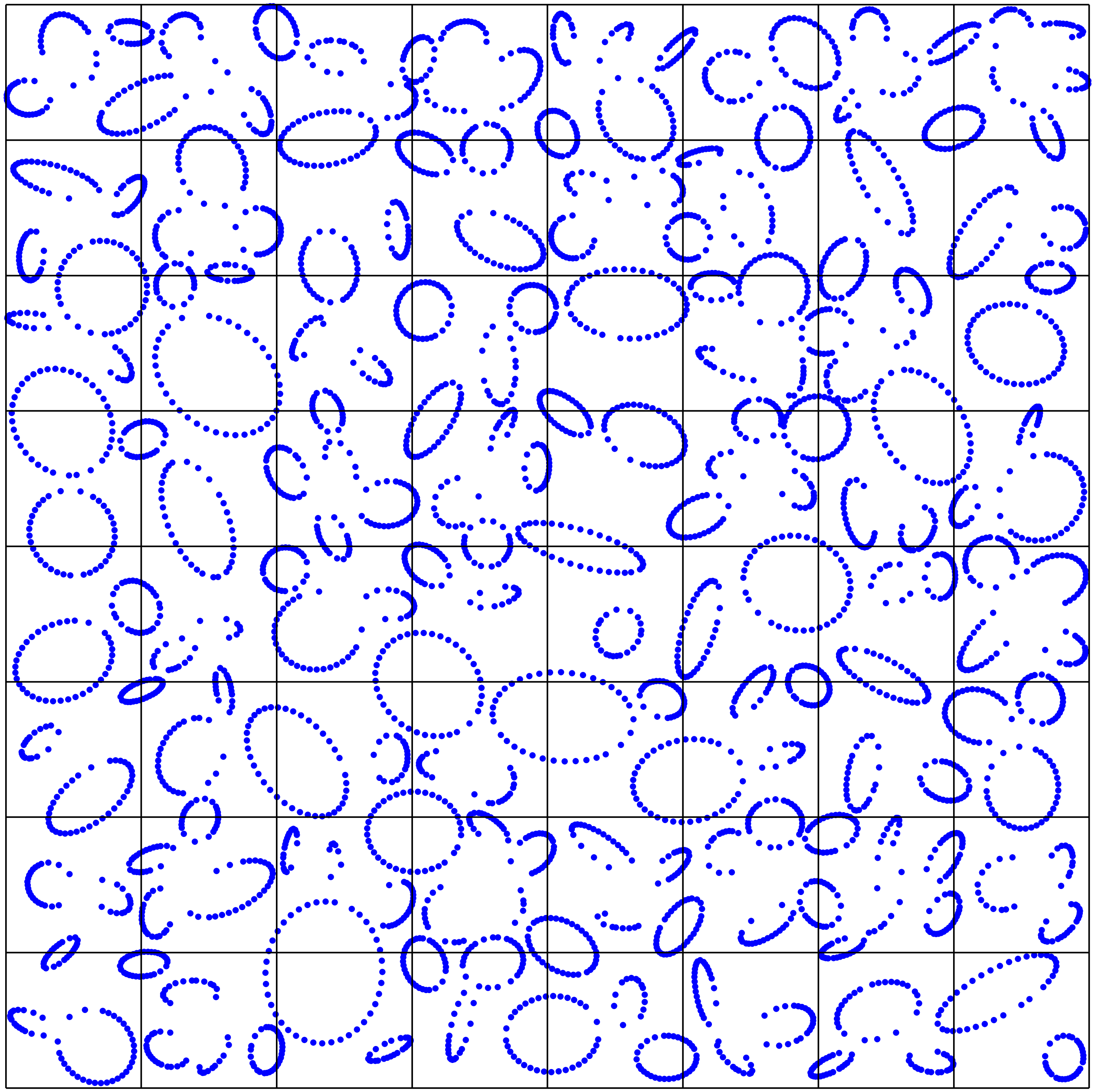
Let Ω_1 denote the block marked in red.

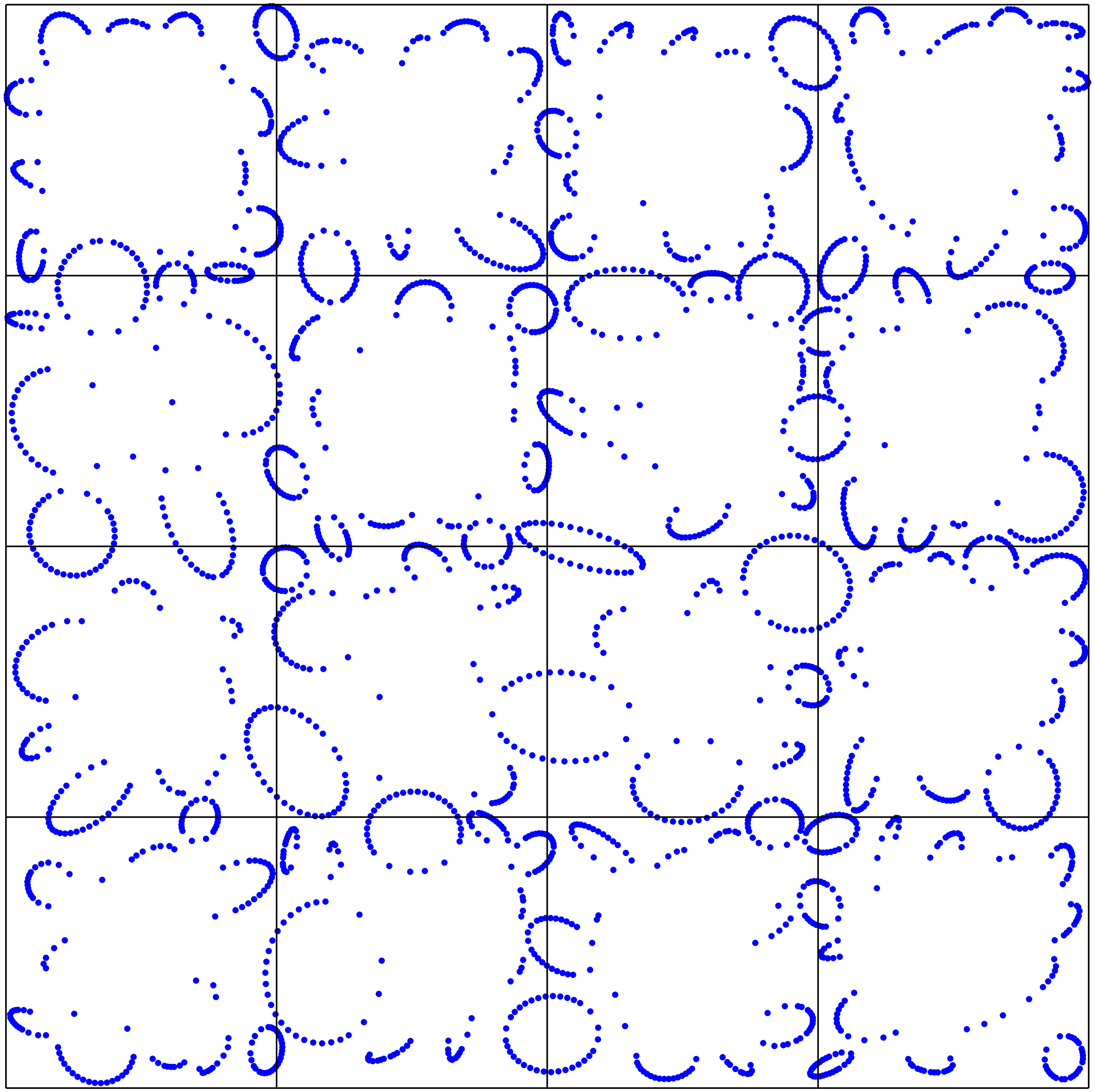
Let Ω_2 denote the rest of the domain.

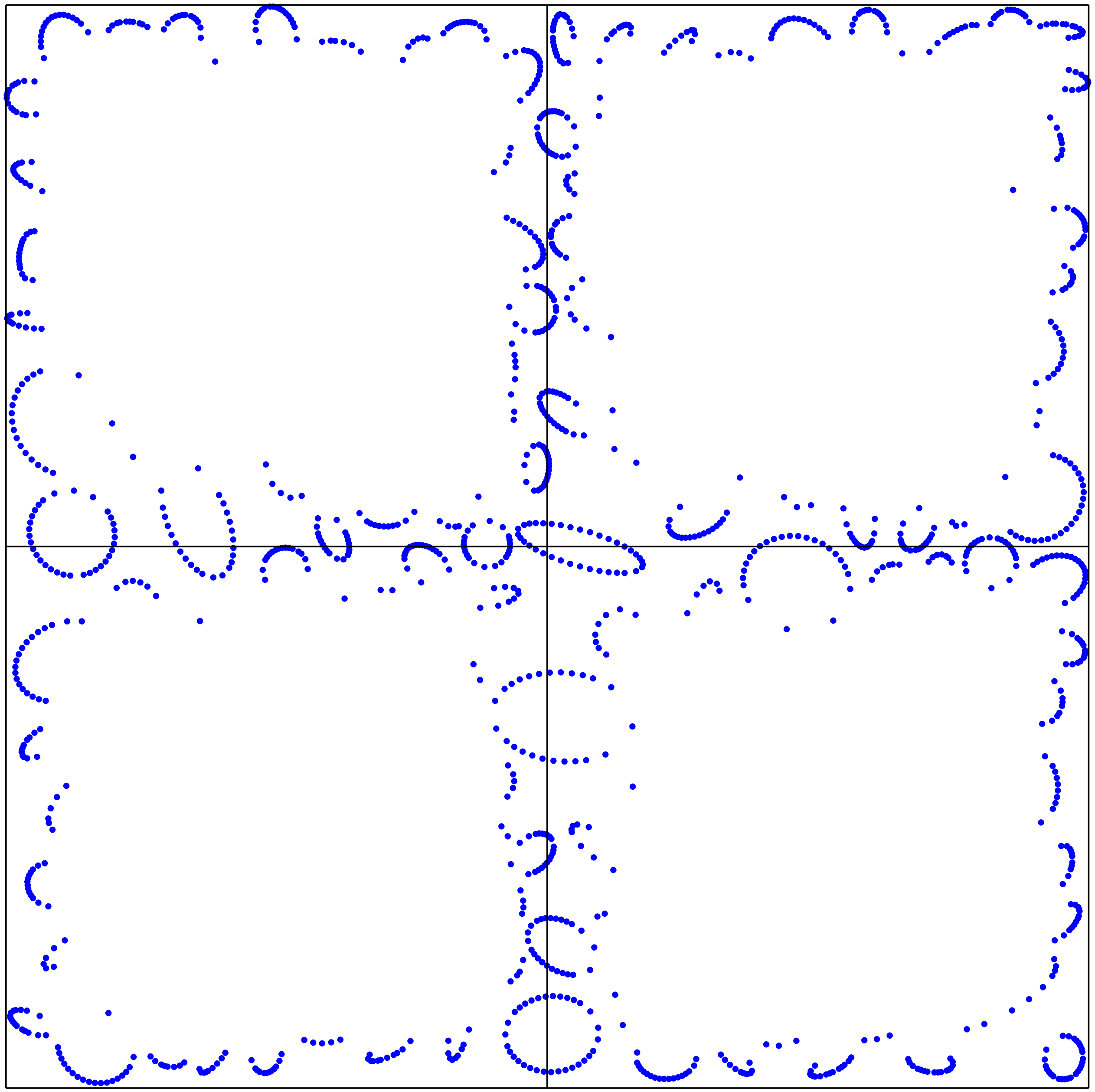
$$\begin{tikzcd}
\text{Charges on } \Omega_2 \arrow[r, "A_{12}"] \arrow[dr, "A_{12}^{\rm skel}"'] & \text{Pot. on } \Omega_1 \arrow[r, "A_{11}^{-1}"] & \text{Charges on } \Omega_1 \arrow[r, "A_{21}"] \arrow[d, "U_1^{\rm t}"'] & \text{Pot. on } \Omega_2 \\
 & \text{Pot. on } \Omega_1^{\rm skel} \arrow[u, "U_1"'] \arrow[r, "\tilde{A}_{11}^{-1}"] & \text{Charges on } \Omega_1^{\rm skel} \arrow[ur, "A_{12}^{\rm skel}"'] &
\end{tikzcd}$$

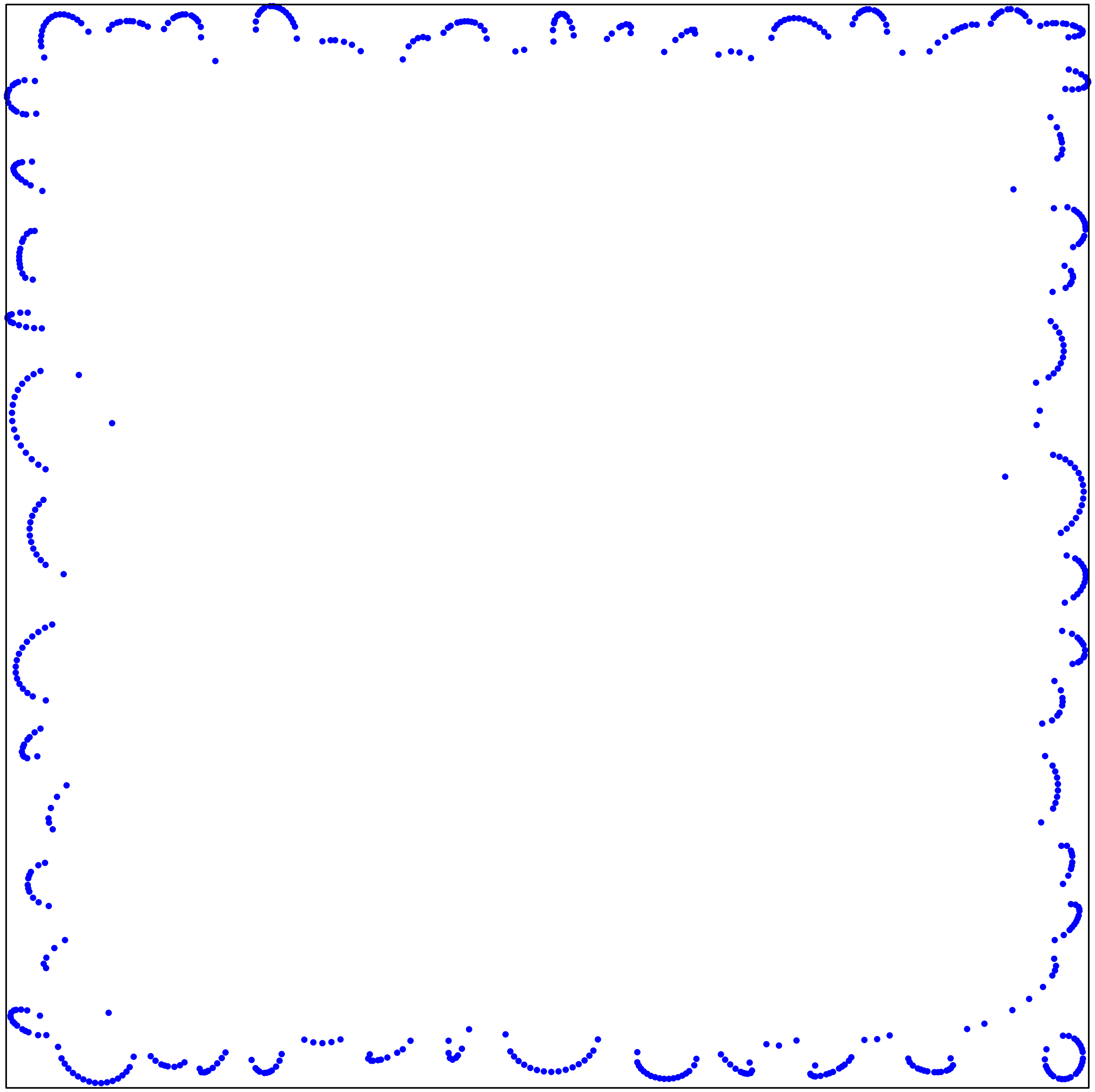
$\tilde{A}_{11}$ contains *all the information the outside world needs to know about* Ω_1.

To obtain a globally $O(N)$ scheme, we hierarchically merge proxy matrices.









Let us return to the question of how to compute a “proxy matrix”.

$$\begin{tikzcd}
\text{Charges on } \Omega_2 \arrow[r, "A_{12}"] \arrow[dr, "A_{12}^{\text{skel}}"'] & \text{Pot. on } \Omega_1 \arrow[r, "A_{11}^{-1}"] & \text{Charges on } \Omega_1 \arrow[r, "A_{21}"] \arrow[d, "U_1^{\text{t}}"'] & \text{Pot. on } \Omega_2 \\
& \text{Pot. on } \Omega_1^{\text{skel}} \arrow[u, "U_1"'] \arrow[r, "\tilde{A}_{11}^{-1}"] & \text{Charges on } \Omega_1^{\text{skel}} \arrow[ur, "A_{12}^{\text{skel}}"'] &
\end{tikzcd}$$

Computing $\tilde{A}_{11}$ amounts to constructing a low-rank approximation

$$A_{21}\,A_{11}^{-1}\,A_{12} \approx A_{21}^{\text{skel}}\,\tilde{A}_{11}^{-1}\,A_{12}^{\text{skel}}.$$

Note that all the matrices A_{ij} can be applied cheaply!

Randomized algorithms:

(Mark Tygert, Vladimir Rokhlin, PGM)

A rank-k approximation to a matrix A can be obtained via the application of A to $k + 20$ random vectors.

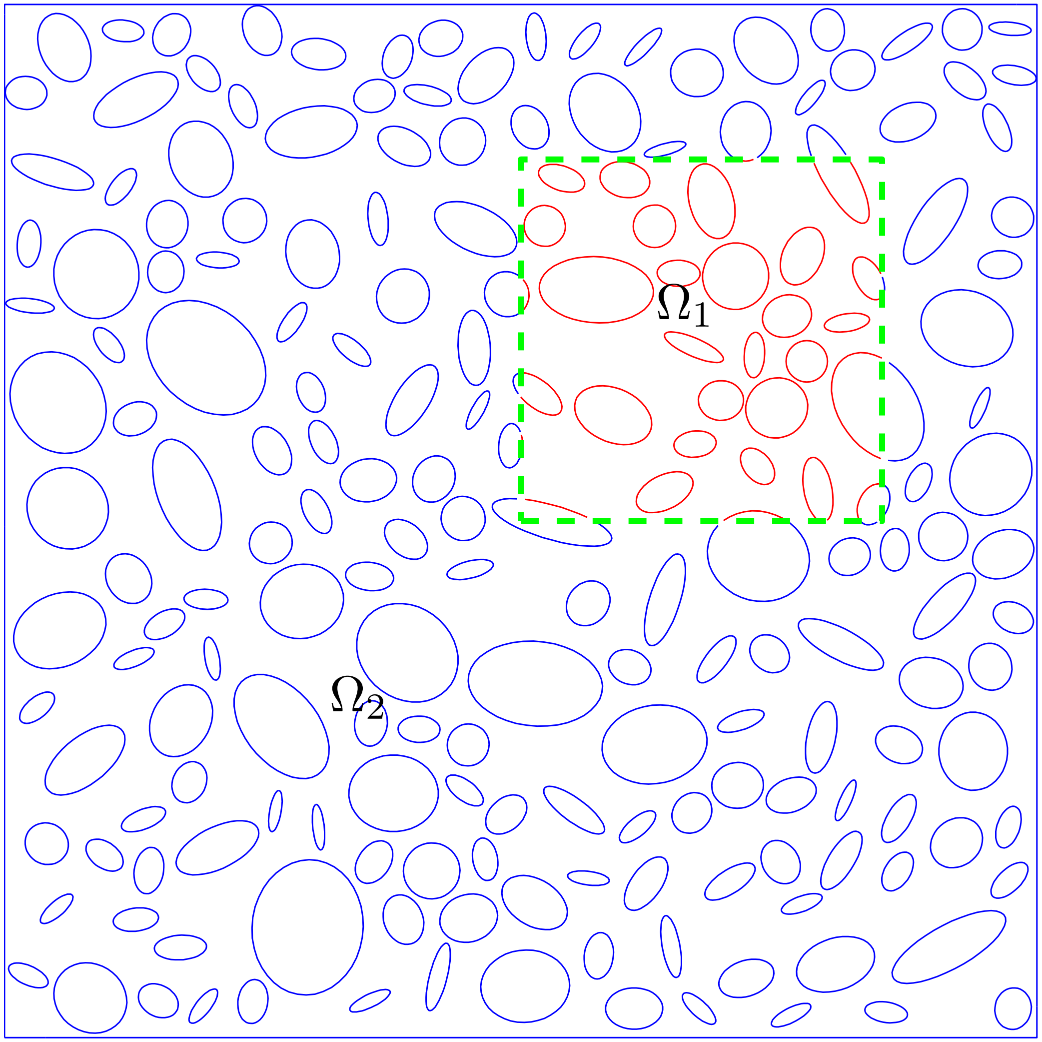
Theorem: *Let A be an $m \times n$ matrix and let k be an integer. Set $l = k + 20$, and let G be an $n \times l$ matrix with i.i.d. Gaussian elements. Let $(q_j)_{j=1}^k$ denote the first k left singular vectors of $A\,G$ and set $Q = [q_1, \dots, q_k]$. Then with probability at most 10^{-17},*

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\ \sqrt{(k+20)\,m}\ \sigma_{k+1},$$

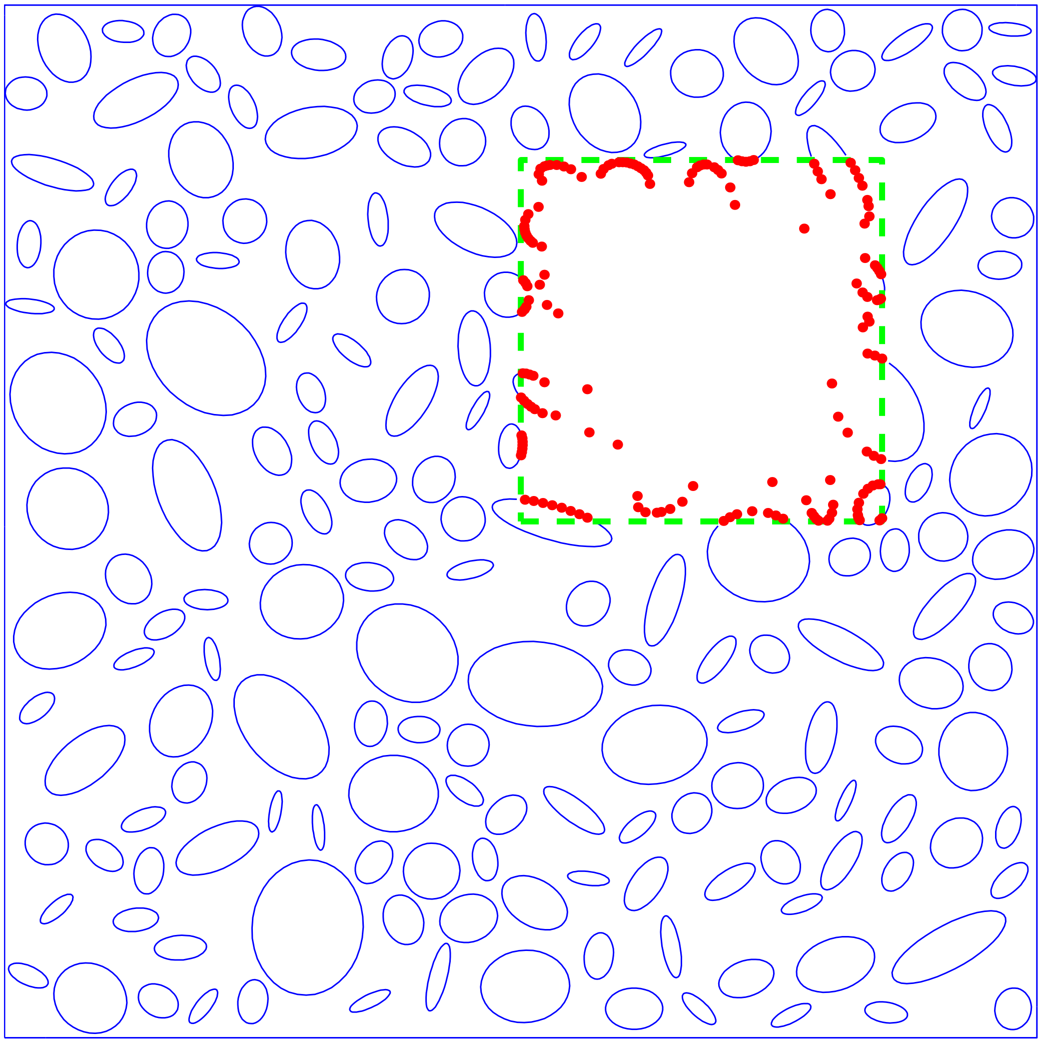
where σ_j denotes the j'th singular value of A.

Particularly useful when a fast method for matrix-vector multiplies is available (FFT, sparse matrices, FMM). It appears to be more robust than Krylov subspace methods.

Example: How do we compute the proxy matrix for:



We seek to compress the map $A_{21}\, A_{11}^{-1}\, A_{12}$.



There is an obvious connection to multi-scale modelling.

The proxy matrix is a reduced model for the piece of material.

Benefits of using proxy matrices for model reduction ("coarse-graining"):

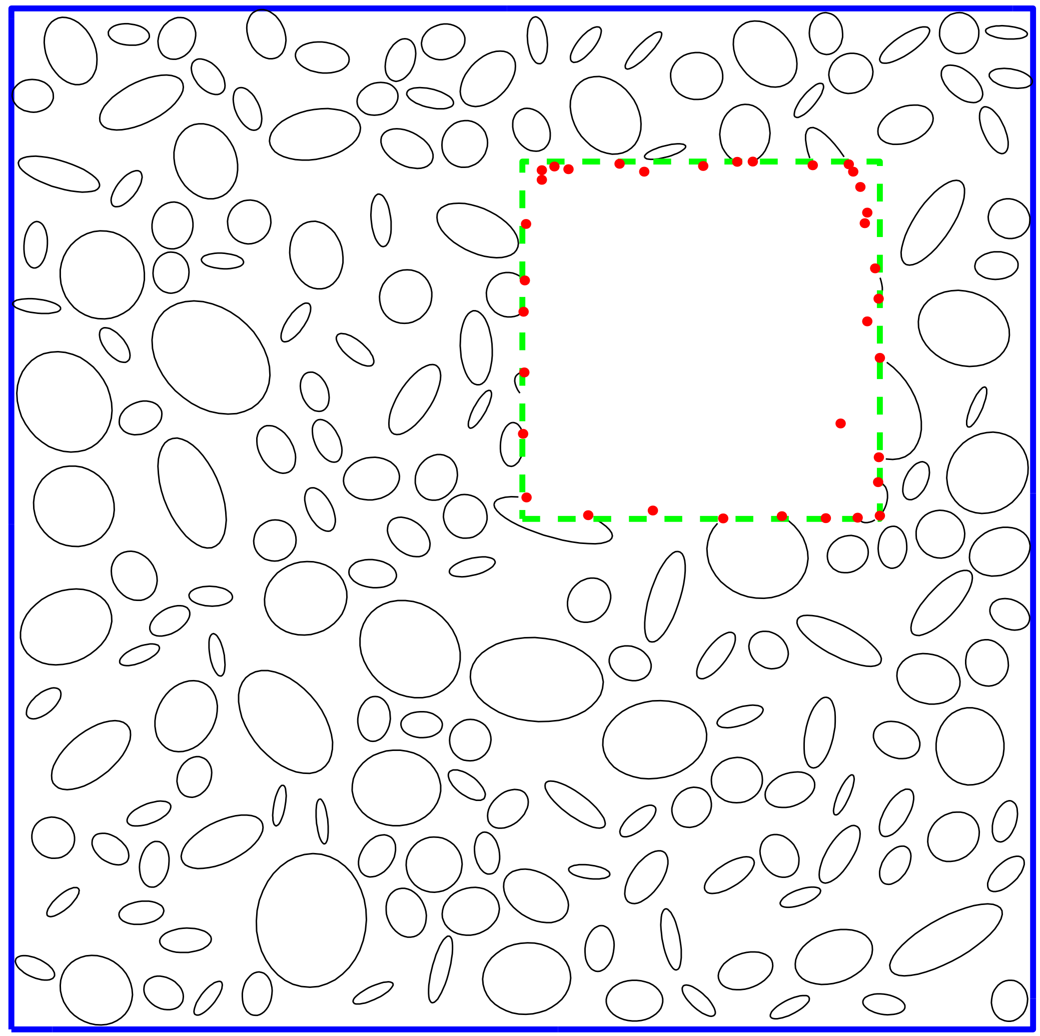
- It works for non-smooth loads.
- It works near boundaries.
- It does not rely on specific macro-scopic behavior (*e.g.* ellipticity does not need to be preserved).
- The proxy matrices make excellent "building blocks" – it is easy to join them together. No need for "buffer zones". Libraries of model reduced patches (constructed from random micro-structures) can be assembled.

Heuristically speaking; this method of constructing reduced models consists of approximating the *inverse* of an operator, rather than the operator itself.

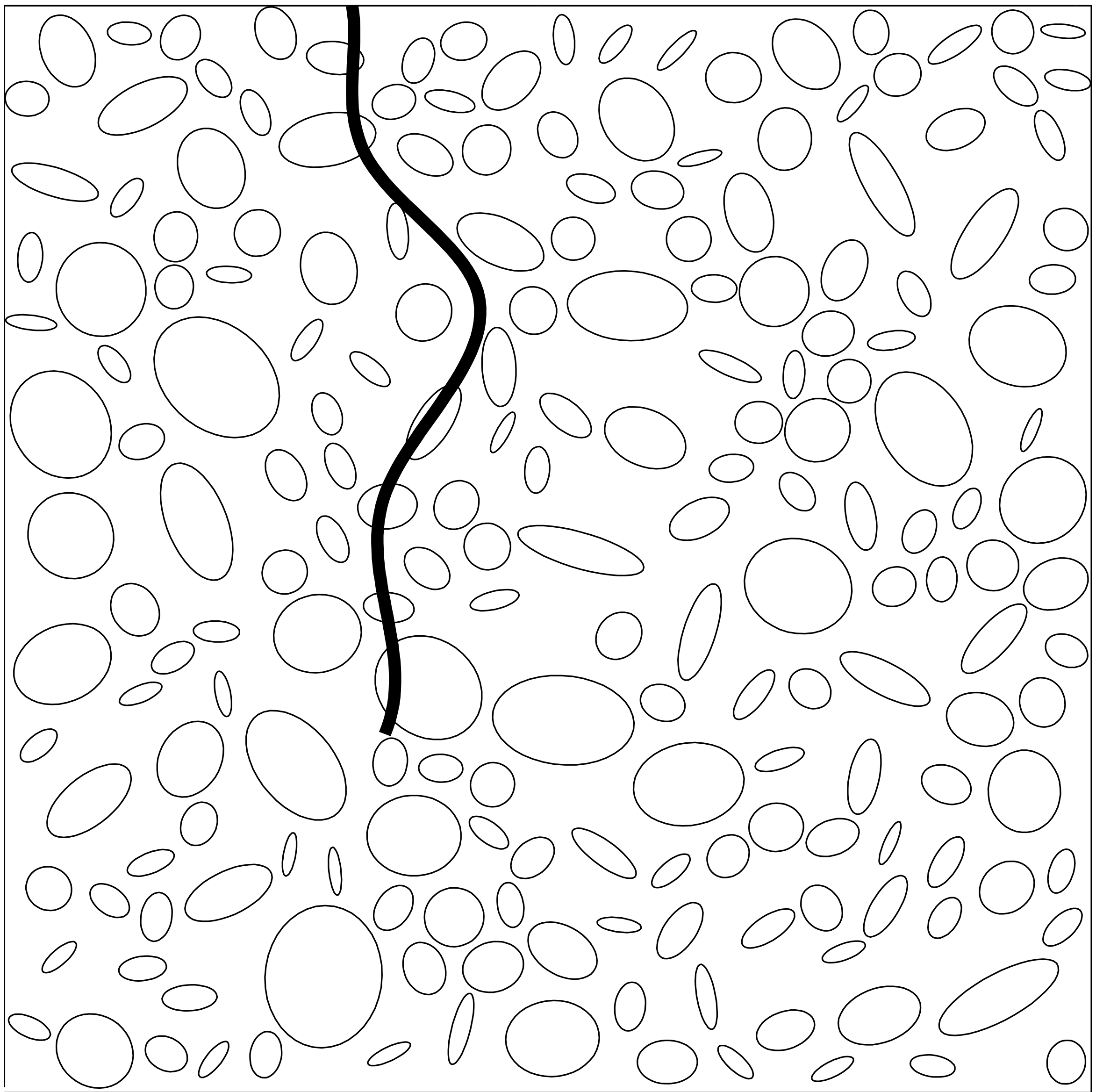
If, however, the region to be compressed *is* well-separated from the boundary, and not subjected to sharp loads, then the direct method is capable of taking full advantage of the simplifications.

In this example, the piece in the middle interacts *with the boundary only*.

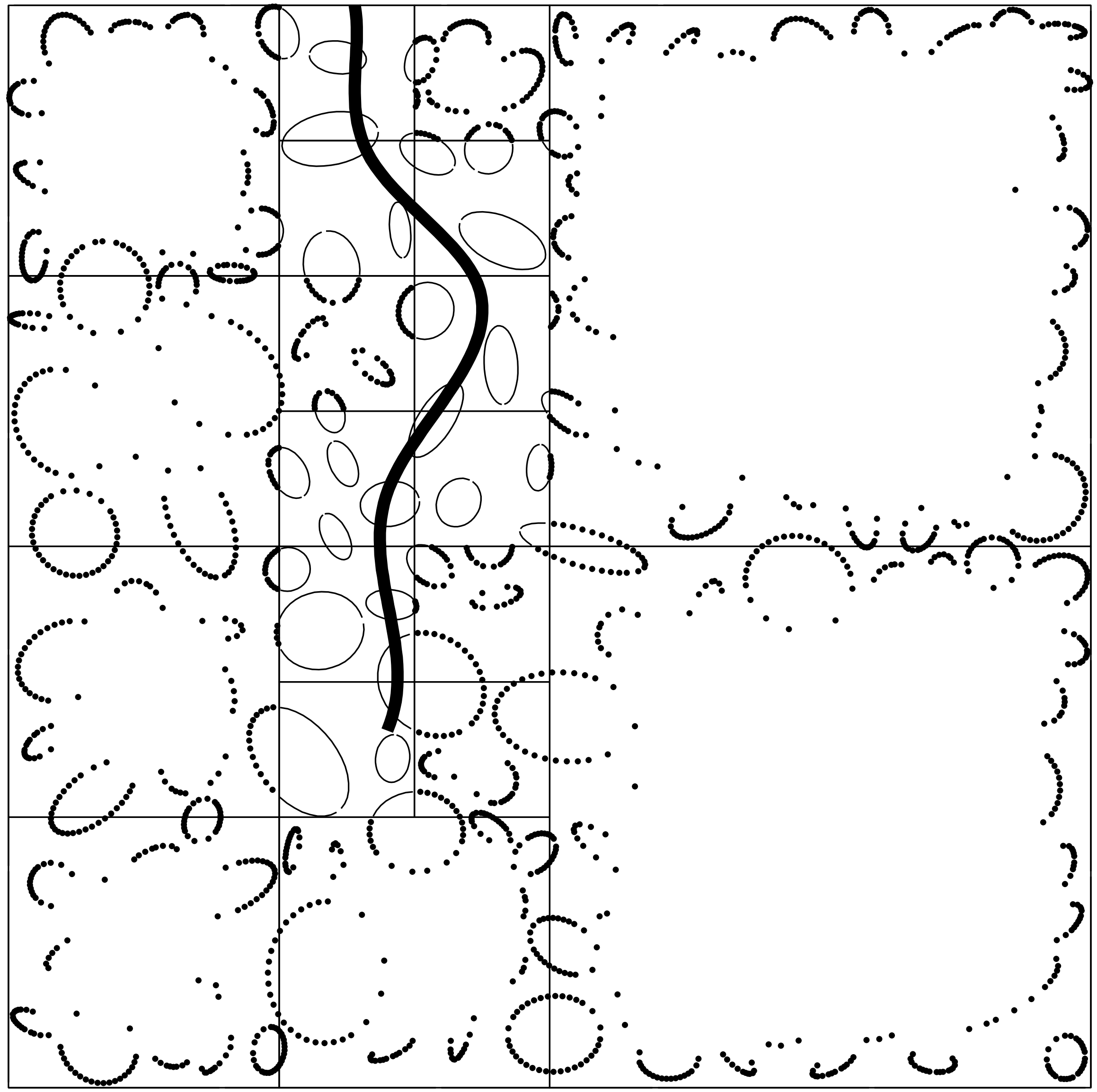
The rank is then reduced from 167 to 35 (at $\varepsilon = 10^{-8}$).



A crack propagating through a composite material:

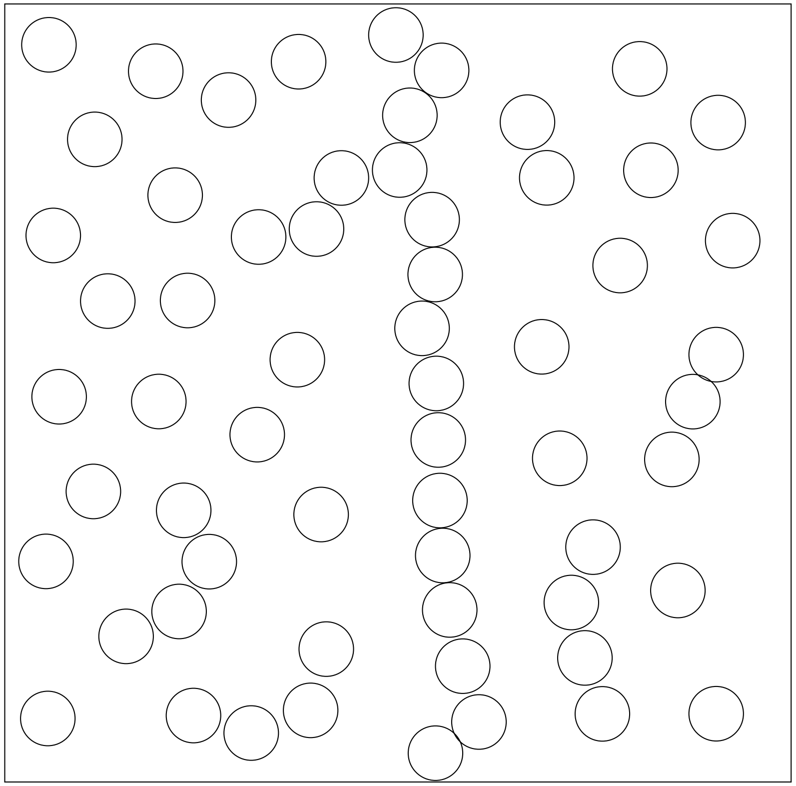


Any unaffected part of the material can be replaced by proxies.



The "direct" model-reduction technique works in environments that are challenging to existing techniques; for instance

- Composite materials near boundaries.
- Percolation problems.
- Wave propagation problems.



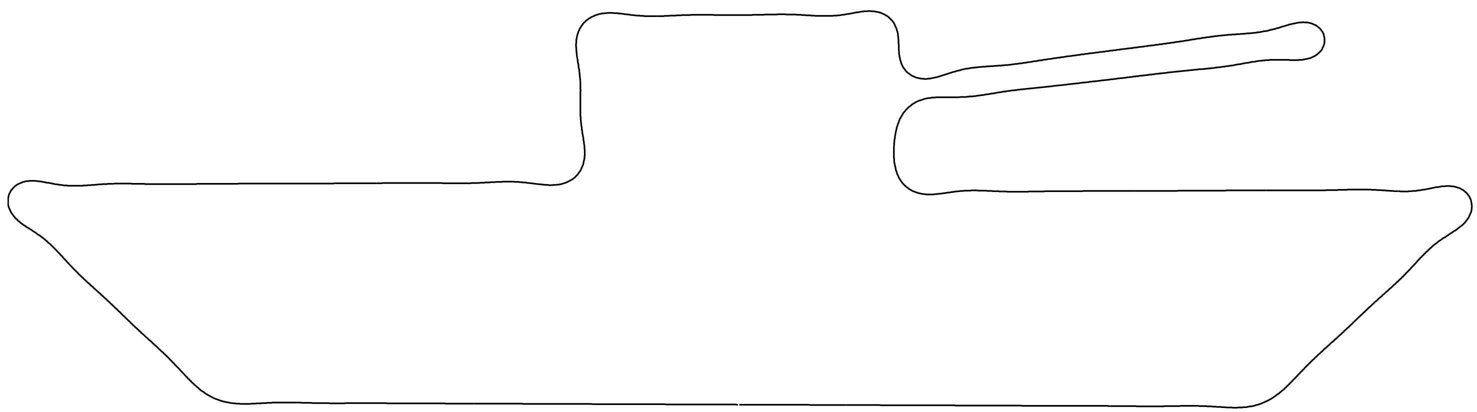
A "percolating" micro-structure.

Numerical examples

In developing direct solvers, the “proof is in the pudding” — recall that from a theoretical point of view, the problem is already solved (by Hackbusch and others).

All computations were performed on standard laptops and desktop computers in the 2.0GHz - 2.8Ghz speed range, and with less than 1Gb of RAM.

An exterior Helmholtz Dirichlet problem



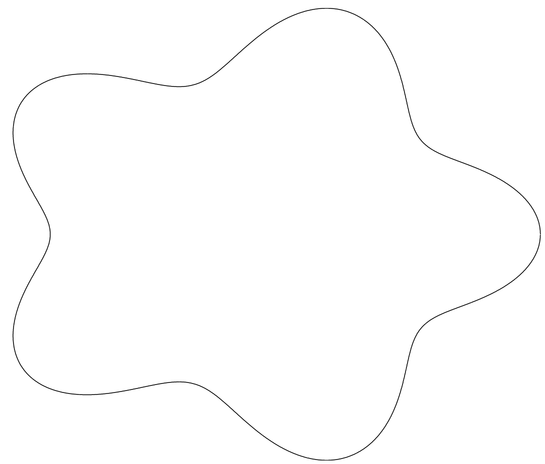
A smooth contour. Its length is roughly 15 and its horizontal width is 2.

k	N_{start}	N_{final}	t_{tot}	t_{solve}	E_{res}	E_{pot}	$\sigma_{\min}$	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 10^{th} 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 $O(n + k^3)$.

Example 2 - An interior Helmholtz Dirichlet problem

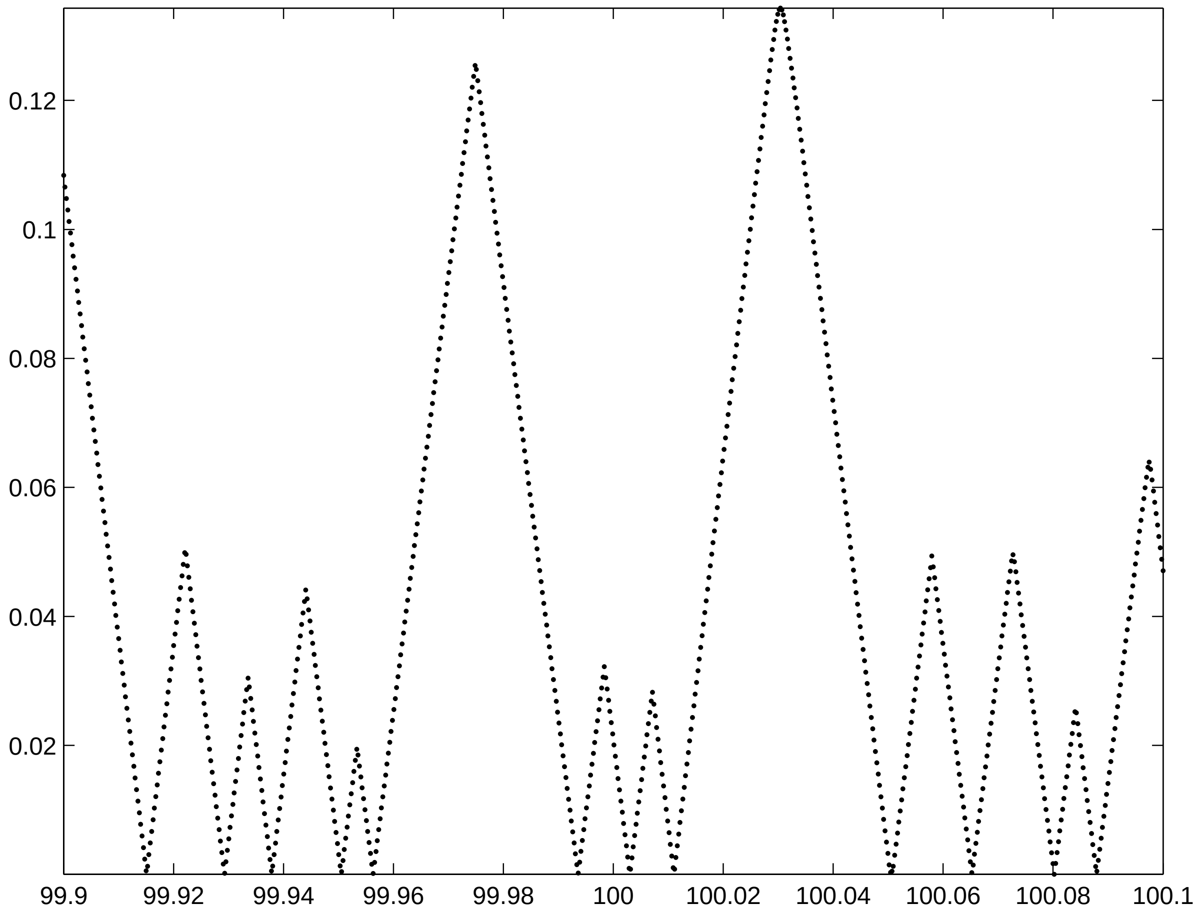


The diameter of the contour is about 2.5. An interior Helmholtz problem with Dirichlet boundary data was solved using $N = 6\,400$ discretization points, with a prescribed accuracy of 10^{-10}.

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

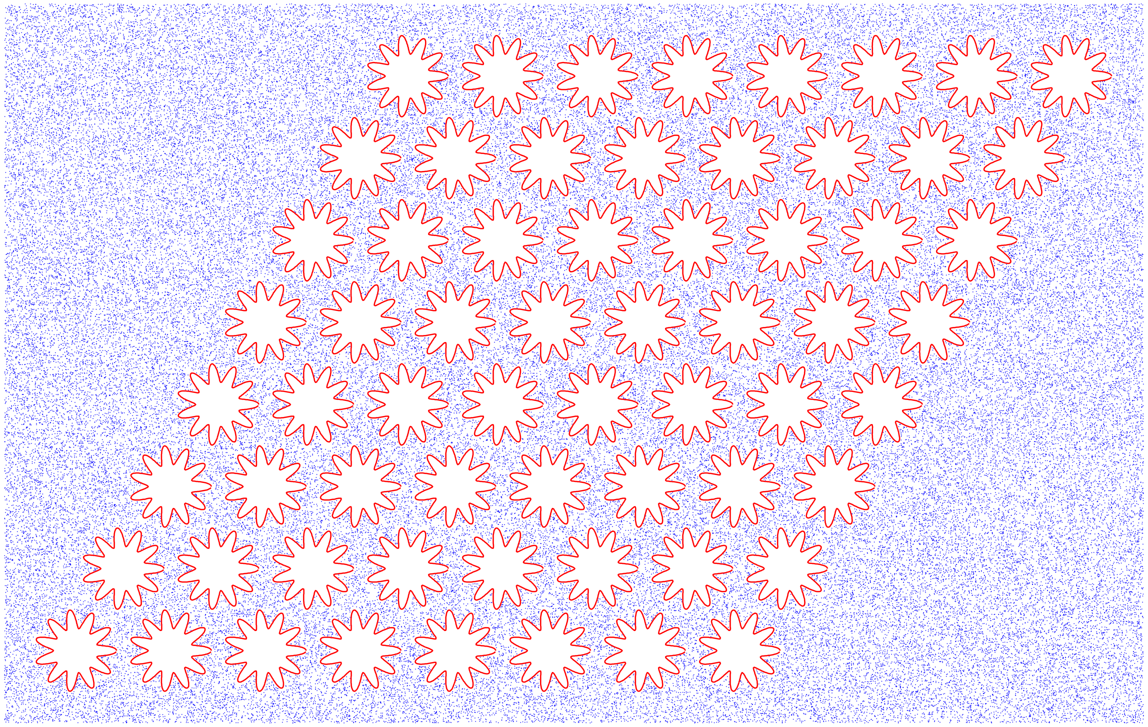
Time for constructing the inverse: 0.7 seconds.

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.

An electrostatics problem in a dielectrically heterogeneous medium

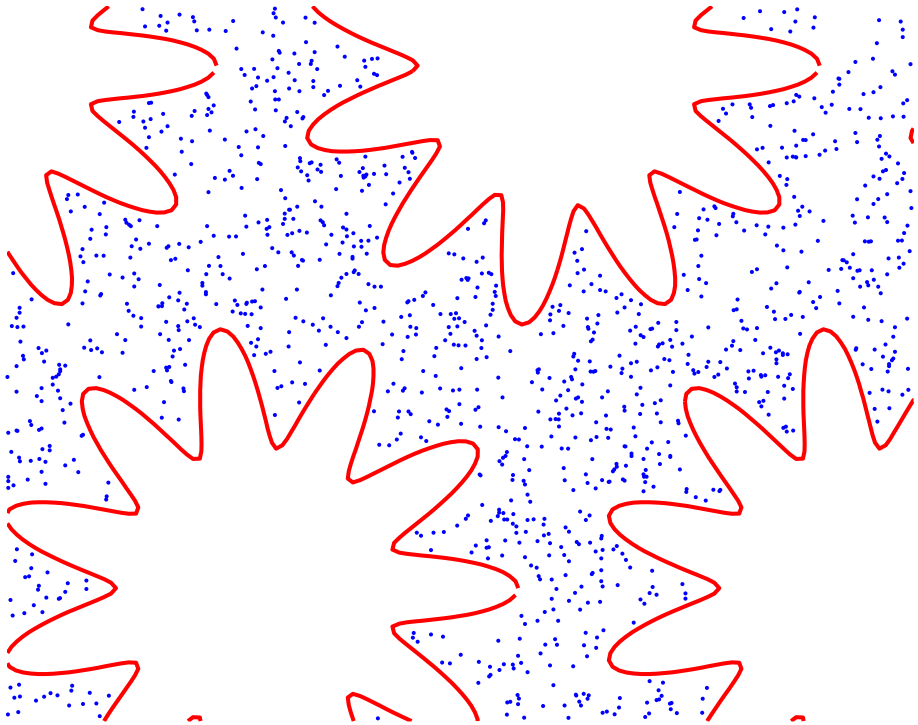


$\varepsilon = 10^{-5}$ $\qquad N_{\text{contour}} = 25\,600$ $\qquad N_{\text{particles}} = 100\,000$

Time to invert the boundary integral equation = 46sec.

Time to compute the induced charges = 0.42sec.(2.5sec for the FMM)

Total time for the electro-statics problem = 3.8sec.



A close-up of the particle distribution.

$O(N)$ direct solvers:

- Direct inversion of integral and differential operators.
- Very stable, relatively insensitive to conditioning. Multiple solves.
- Have proven capable of solving previously intractable problems in 2D.

Randomized algorithms:

- Applications to direct solvers and multiscale modeling.
- Alternative to Lansczos-methods. Appears to be much more stable.
- Applications to network matrices, *e.g.* search algorithms.

Multiscale modelling:

- Approximation of the *inverses* of difference operators.
- No smoothness assumptions; can handle boundaries.
- Wave propagation problems. Percolation. Crack-propagation.

An interpolation result (Rokhlin/Tygert/PGM):

Theorem: *Let Ω be a compact set, and let V be a k-dimensional space of complex-valued continuous functions on Ω.*

Then there exist k points $\{x_j\}_{j=1}^{k} \subseteq \Omega$, and k continuous functions $\{\varphi_j\}_{j=1}^{k}$ on Ω such that for all $f \in V$,

$$f(x) = \sum_{j=1}^{k} f(x_j)\,\varphi_j(x).$$

Moreover, for $j = 1, \dots, k$,

$$|\varphi_j(x)| \leq 1, \quad \forall\, x \in \Omega.$$

In many environments, the points $(x_n)_{n=1}^{N}$ and the functions $(\varphi_n)_{n=1}^{N}$ can be computed rapidly and accurately.

For a result regarding an $m \times n$ matrix A of rank k, simply set:

$$\Omega = \{1,\, 2,\, \dots,\, n\}, \qquad V = \mathrm{Row}(A).$$