Analysis based methods for solving linear elliptic PDEs numerically

Gunnar Martinsson

Students & postdocs: Tracy Babb, Anna Broido, Nathan Halko, Sijia Hao, Nathan Heavner, Adrianna Gillman, Dan Kaslovsky, Sergey Voronin, Patrick Young.

Research support by:

Problem addressed: How do you efficiently compute approximate solutions to linear boundary value problems (BVPs) of the form

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

where Ω is a domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary Γ, and where A is an elliptic differential operator (constant coefficient, or not). Examples include:

- The Laplace equation.
- 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:

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

Problem addressed: How do you efficiently compute approximate solutions to linear boundary value problems (BVPs) of the form

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

where Ω is a domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary Γ, and where A is an elliptic differential operator (constant coefficient, or not).

Observation: The problem is *in principle* easy to solve! Simply integrate

$$\text{(SLN)} \qquad u(\boldsymbol{x}) = \int_\Omega G(\boldsymbol{x},\boldsymbol{y})\,g(\boldsymbol{y})\,d\boldsymbol{y} + \int_\Gamma F(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,dS(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Omega,$$

where G and F are two kernel functions that depend on A, B, and Ω.

Good: The operators in (SLN) are generally "nice" — bounded, smoothing, etc.

Problem 1: The operators in (SLN) are *rarely known analytically.*

Problem 2: The operators in (SLN) are *global.* $\rightarrow$ Dense matrices upon discretization.
$\rightarrow$ Computationally expensive.

Well-established results: In simple situations where the solution operator is known analytically, there are mature methodologies for applying the global operators efficiently.

Example: Consider the Poisson equation on $\Omega = \mathbb{R}^2$:

$$-\Delta u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^2,$$

coupled with an appropriate decay condition at infinity for well-posedness.

The solution is given by

$$u(\boldsymbol{x}) = [\phi * g](\boldsymbol{x}) = \int_{\mathbb{R}^2} \phi(\boldsymbol{x} - \boldsymbol{y})\, g(\boldsymbol{y})\, d\boldsymbol{y}, \qquad \boldsymbol{x} \in \mathbb{R}^2, \tag{4}$$

where ϕ is the fundamental solution of the Laplace operator

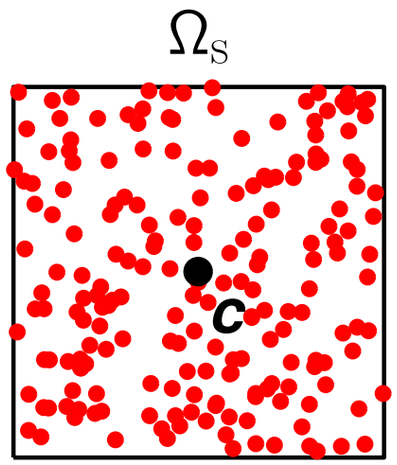
$$\phi(\boldsymbol{x}) = -\frac{1}{2\pi} \log |\boldsymbol{x}|.$$

In order to evaluate (4) numerically, we introduce a set of discretization nodes $\{\boldsymbol{x}_i\}_{i=1}^N$ in the support of g. Then upon application of a quadrature rule (some difficulties arise, since the kernel is singular ...) one is left with the task of evaluating sums of the type

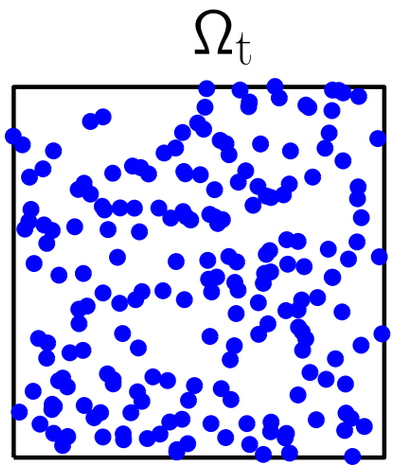
$$u_i = \sum_{j \neq i} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1, 2, \ldots, N.$$

Remark: Alternatively, one could to move to Fourier space using the FFT, invert the resulting algebraic equation $|\boldsymbol{t}|^2 \hat{u}(\boldsymbol{t}) = \hat{f}(\boldsymbol{t})$, then move back to physical space via a second FFT. This is *very* fast, but the approach has somewhat limited applicability (uniform grid, periodic BC, moderate accuracy, etc.).

The Fast Multipole Method (FMM)



Sources q_j at locations $\boldsymbol{y}_j$.



Potentials u_i at locations $\boldsymbol{x}_i$.

Consider the task of evaluating the sum

$$(1) \qquad w_i = \sum_{j=1}^{N} \log(\boldsymbol{x}_i - \boldsymbol{y}_j)\, q_j,$$

for $i = 1, 2, \ldots, M$,

representing evaluation of potentials u_i at locations $\{\boldsymbol{x}_i\}_{i=1}^{M}$ resulting from sources q_j at locations $\{\boldsymbol{y}_j\}_{j=1}^{N}$. Observe first that the kernel $\log(\boldsymbol{z})$ admits a separation of variables

$$(2) \qquad \log(\boldsymbol{x} - \boldsymbol{y}) = \log(\boldsymbol{x} - \boldsymbol{c})1 + \sum_{p=1}^{\infty} \frac{-1}{p} \frac{1}{(\boldsymbol{x} - \boldsymbol{c})^p} (\boldsymbol{y} - \boldsymbol{c})^p,$$

where $\boldsymbol{c}$ is the center of Ω_s. Truncate (2) and insert the result into (1)

$$w_i \approx \sum_{j=1}^{N} \left(\log(\boldsymbol{x}_i - \boldsymbol{c})1 + \sum_{p=1}^{P} \frac{-1}{p} \frac{1}{(\boldsymbol{x}_i - \boldsymbol{c})^p} (\boldsymbol{y} - \boldsymbol{c})^p \right) q_j = \log(\boldsymbol{x}_i - \boldsymbol{c})\hat{q}_0 + \sum_{p=1}^{P} \frac{1}{(\boldsymbol{x}_i - \boldsymbol{c})^p} \hat{q}_p$$

where $\hat{q}_0 = \sum_{j}^{n} q_j$ and $\hat{q}_p = -\frac{1}{p} \sum_{j=1}^{N} (\boldsymbol{y}_j - \boldsymbol{c})^p q_j$.

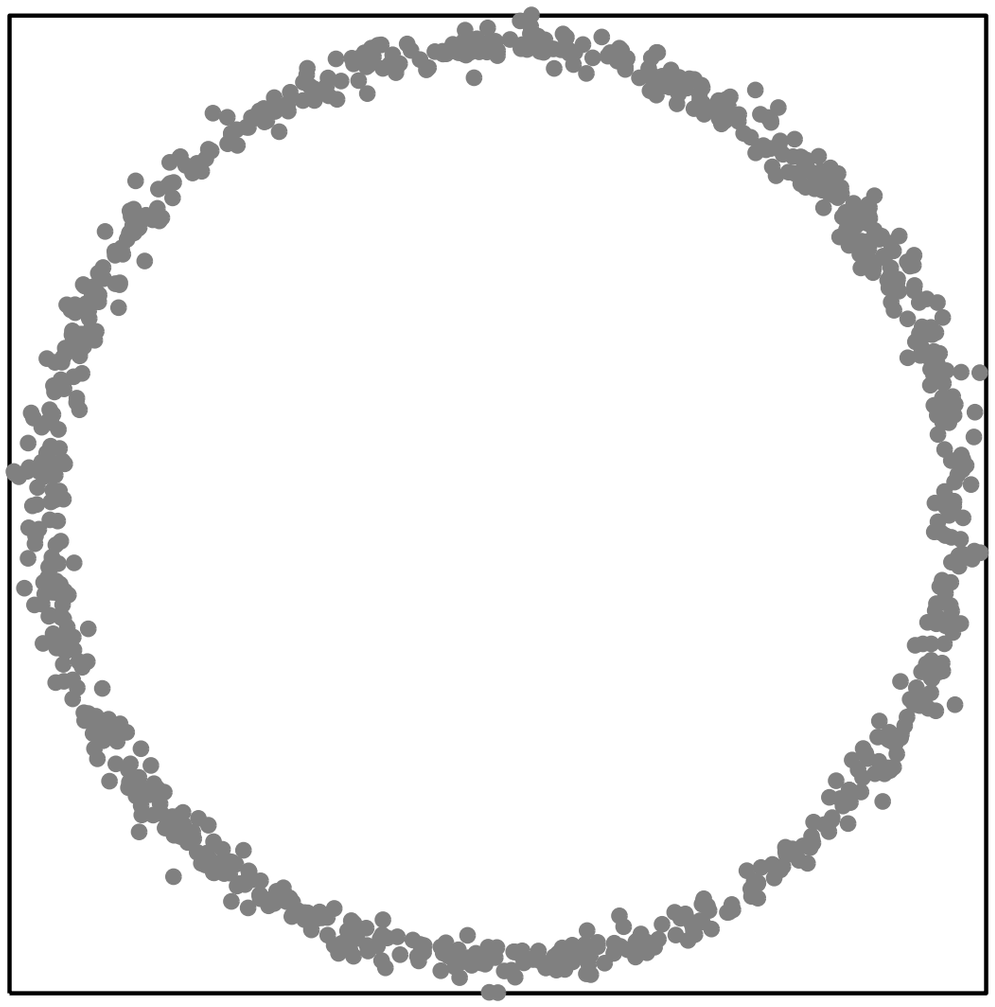
Bottom line: Reduction in cost from $O(MN)$ to $O(P(M+N))$. Error $\sim (\sqrt{2}/3)^P$.

Note: For notational simplicity, we let $\boldsymbol{x}_i, \boldsymbol{y}_j, w_i \in \mathbb{C}$, so the real potential u_i is given by $u_i = Re(w_i)$.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

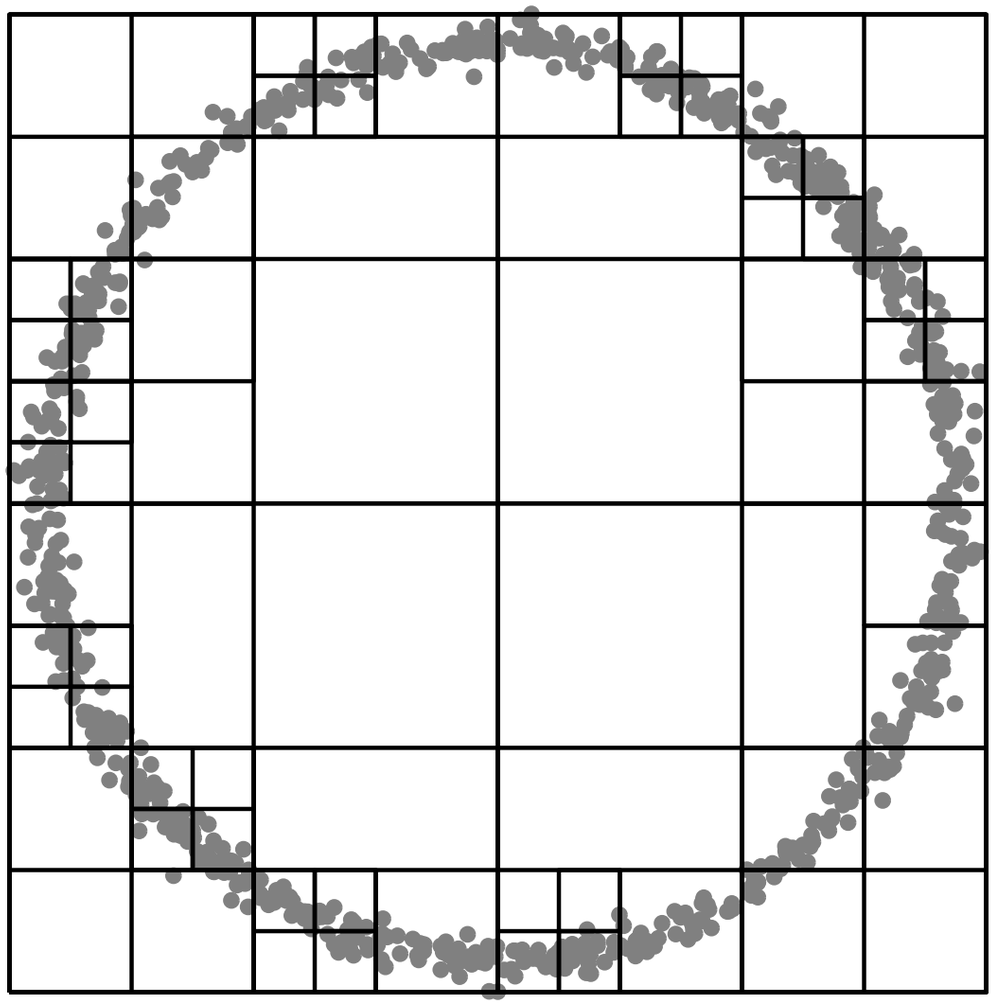


The points $\{\boldsymbol{x}_i\}_{i=1}^N$.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

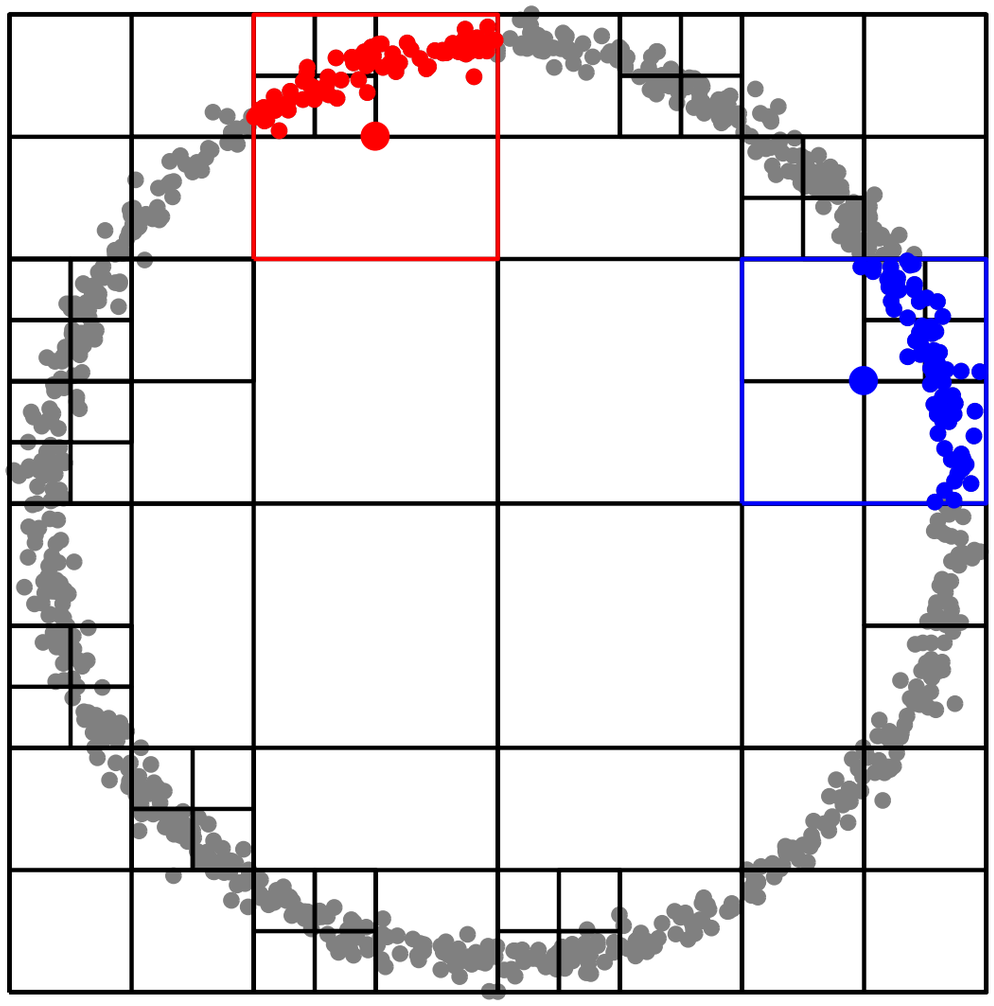


An adaptive quadtree is built so that each leaf holds at most a small number of sources.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

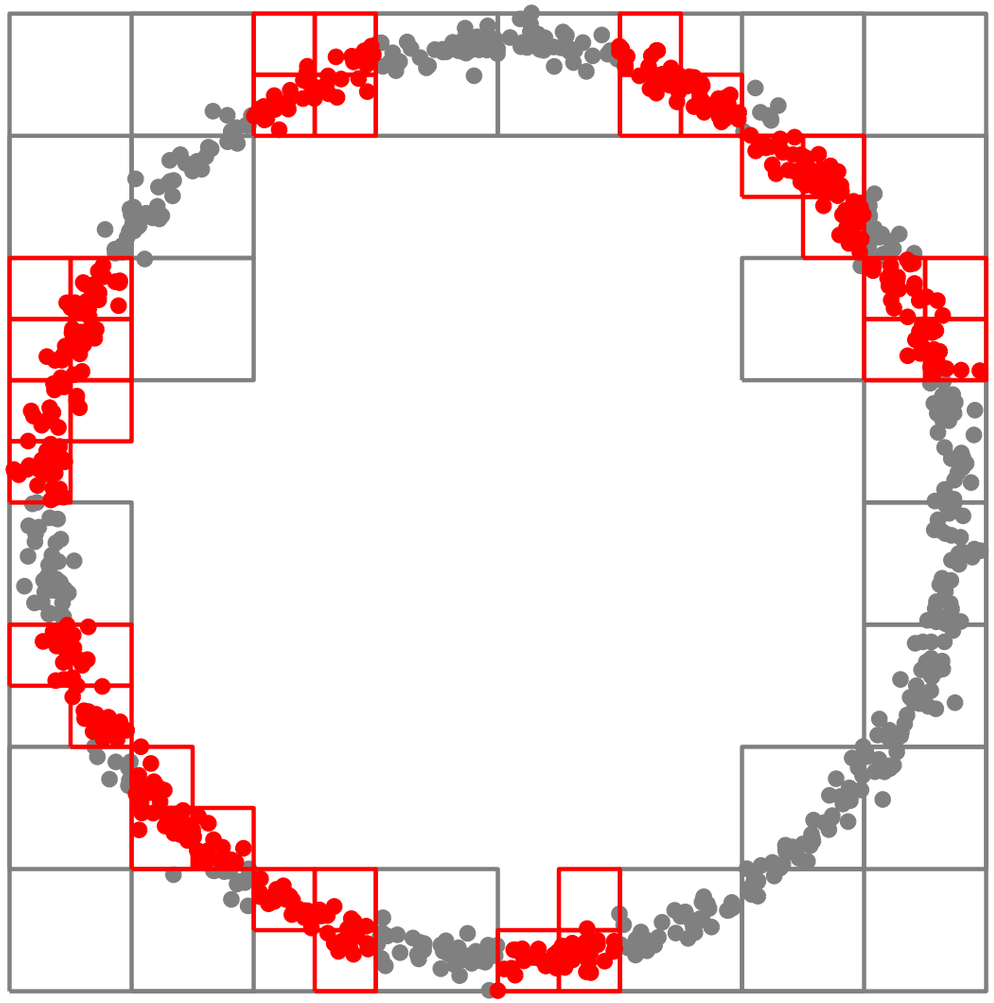


Boxes that are "well-separated" communicate via multiple expansions.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

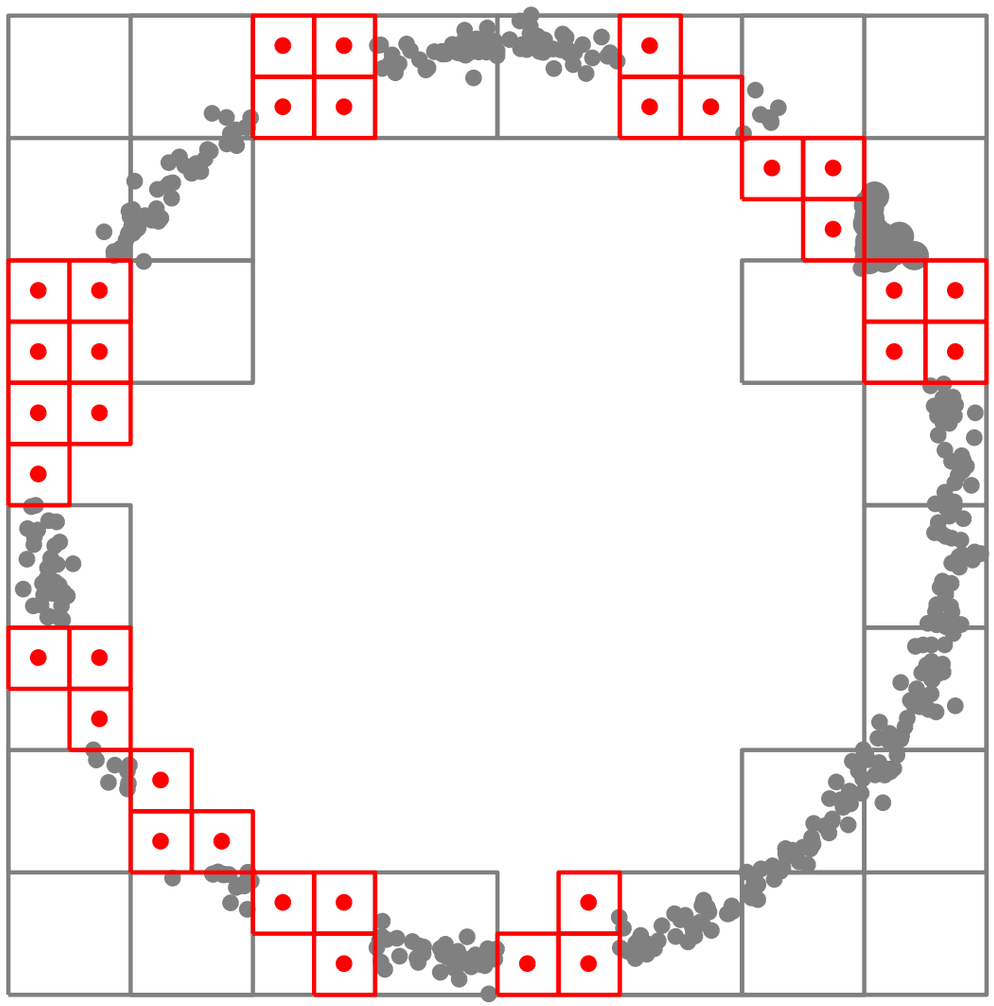


Level 4: Build multipole expansions for the boxes in red.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

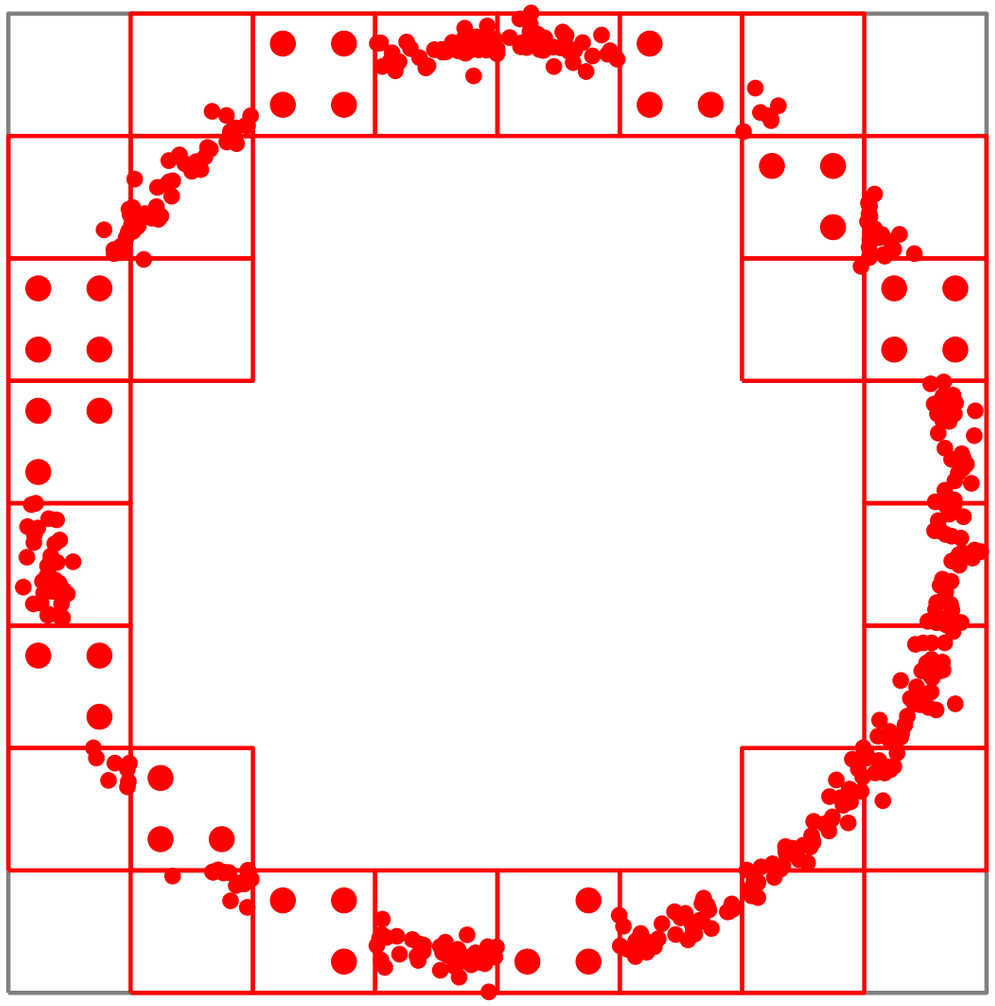


Level 4: Build multipole expansions for the boxes in red.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

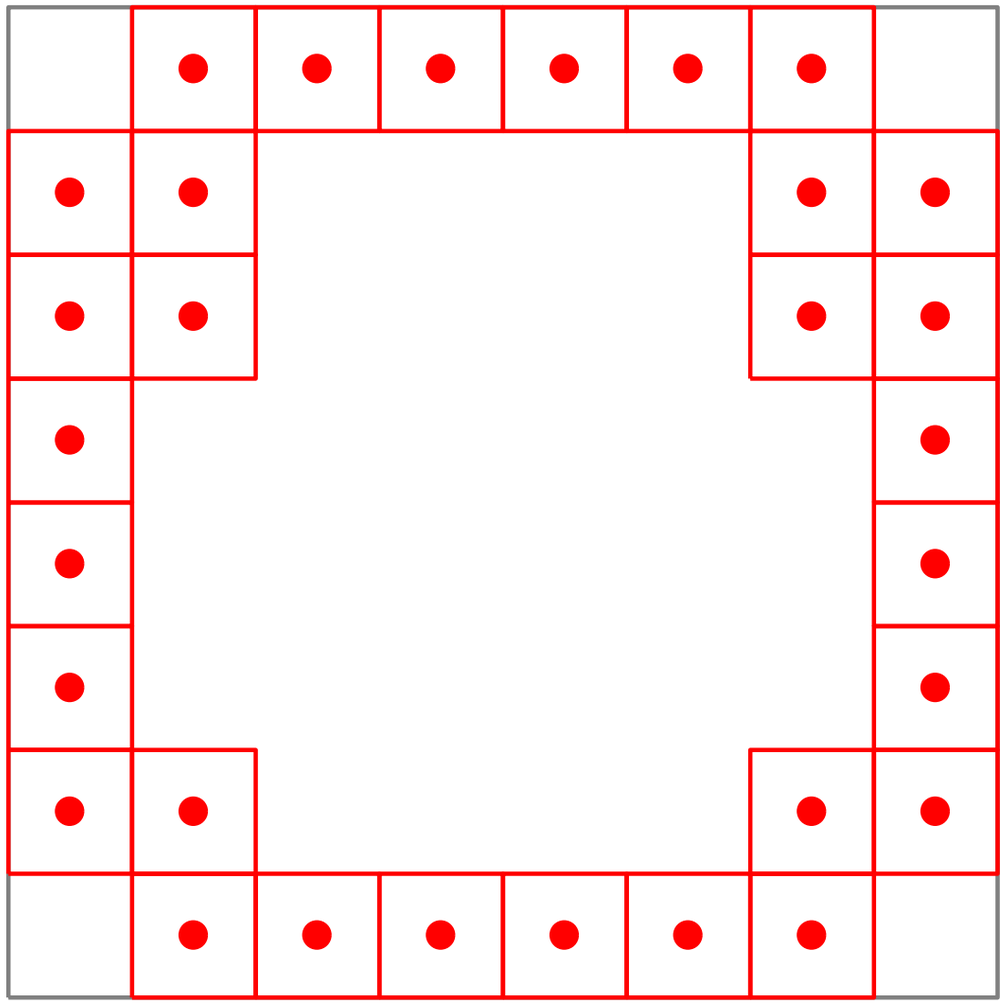


Level 3: Build multipole expansions for the boxes in red.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

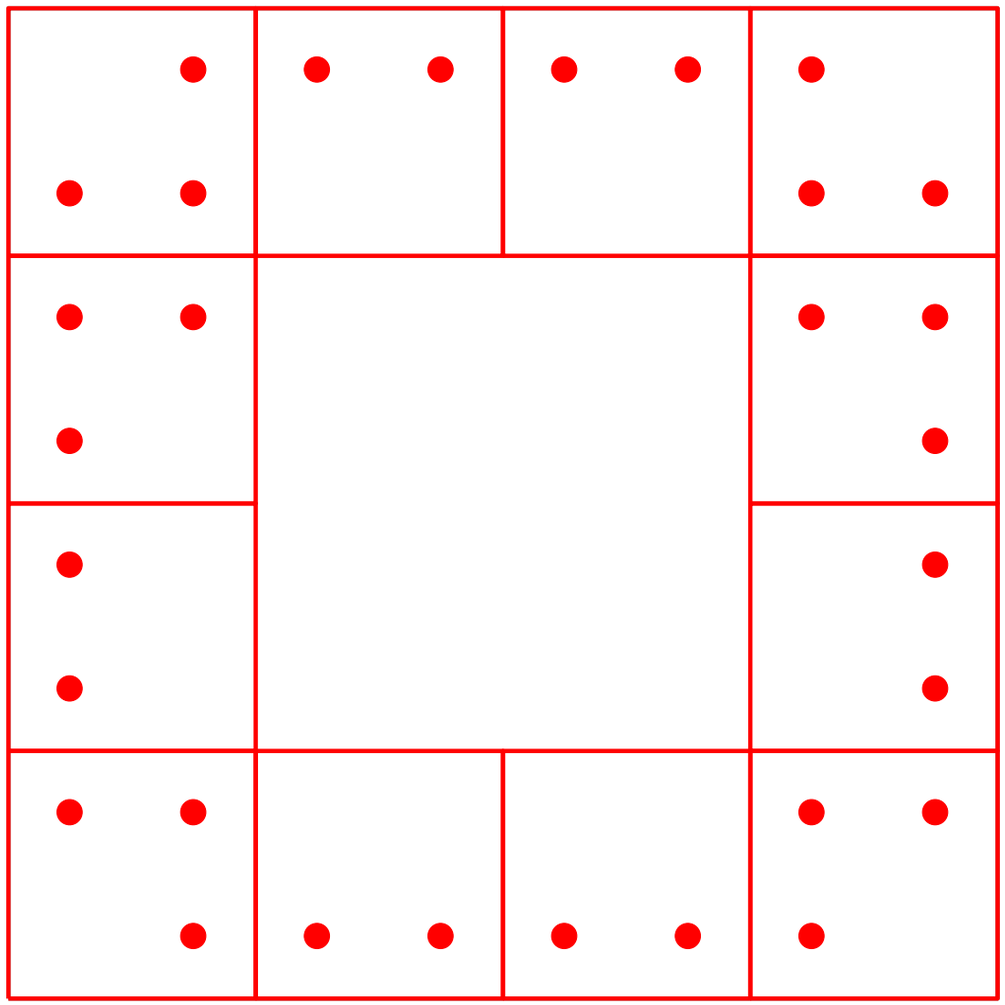


Level 3: Build multipole expansions for the boxes in red.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

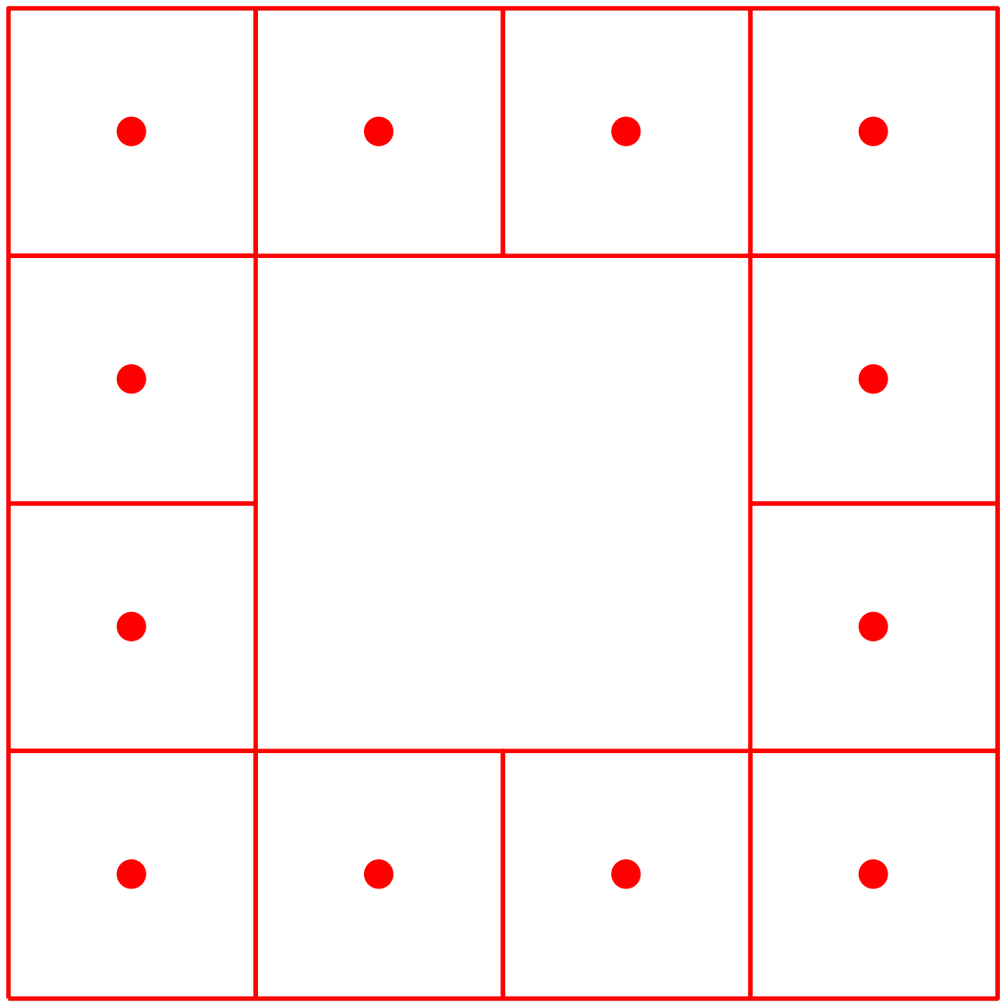


Level 2: Build multipole expansions for the boxes in red.

The Fast Multipole Method (FMM)

The situation discussed on the previous slide was simplistic in that the sources were well separated from the targets. More typically, we are given sets of points $\{\boldsymbol{x}_i\}_{i=1}^N$ and sources $\{q_i\}_{i=1}^N$, and seek to evaluate the sums

$$w_i = \sum_{j=1}^{N} \phi(\boldsymbol{x}_i - \boldsymbol{x}_j)\, q_j, \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$



Level 2: Build multipole expansions for the boxes in red.

Recall that we are interested in solving the PDE $\begin{cases} A\,u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ B\,u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$ (BVP)

Explicit solution formula: $u(\boldsymbol{x}) = \int_\Omega G(\boldsymbol{x},\boldsymbol{y})\,g(\boldsymbol{y})\,d\boldsymbol{y} + \int_\Gamma F(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,dS(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Omega.$ (SLN)

Claim: The global operators in (SLN) are typically amenable to "fast" algorithms analogous to the FMM. These rely crucially on the fact that the dense matrices resulting upon discretization have *off-diagonal blocks of low numerical rank.*

This is a consequence of the *smoothing effect* of elliptic differential equations; it can be interpreted as a *loss of information.*

This effect has many well known physical consequences:

- Rapid convergence of *multipole expansions* when the region of sources is far away from the observation point.
- The *St Venant principle* in mechanics.
- The inaccuracy of imaging at sub-wavelength scales.
- The intractability of solving the heat equation backwards.

Caveat: High-frequency problems present difficulties — no loss of information for length-scales $> \lambda$. Extreme accuracy of optics, high-frequency imaging, *etc*.

Recall that we are interested in solving the PDE $\begin{cases} A\,u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ B\,u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$ (BVP)

Explicit solution formula: $u(\boldsymbol{x}) = \int_\Omega G(\boldsymbol{x},\boldsymbol{y})\,g(\boldsymbol{y})\,d\boldsymbol{y} + \int_\Gamma F(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,dS(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Omega.$ (SLN)

Remaining problem: The kernels F and G are typically not known explicitly.

Standard prescription for solving (BVP) is some variation of:

- Discretize (BVP) directly using Finite Element Method (FEM) / Finite Differences (FD) / ...to obtain a linear system $\mathbf{Au} = \mathbf{b}$.
- Solve the linear system using an iterative solver that generates a sequence of successively better approximations to the exact solution. Each step involves one application of $\mathbf{A}$, which is cheap since $\mathbf{A}$ is sparse.

This paradigm has been extremely successful. Mature and powerful.

Some hairy issues remain:

The matrix $\mathbf{A}$ is necessarily ill-conditioned (since A is unbounded).

For rapid convergence, special "tricks" are required. (Multigrid, pre-conditioners, etc.)

The intermediate problem is nastier than the solution operator. A bit unfortunate ...

Recall that we are interested in solving the PDE $\begin{cases} A\,u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ B\,u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$ (BVP)

Explicit solution formula: $u(\boldsymbol{x}) = \int_{\Omega} G(\boldsymbol{x},\boldsymbol{y})\,g(\boldsymbol{y})\,d\boldsymbol{y} + \int_{\Gamma} F(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,dS(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Omega.$ (SLN)

Core objective of research described: Find numerical methods that directly build an efficient representation of an approximation to the solution operator in (SLN).

- We will draw heavily on results in computational harmonic analysis for efficiently representing certain singular integral operators (the FMM, and its successors).

- We will also draw on existing body of knowledge about finite element methods, finite differences, spectral methods, etc.

- The objective is to find a two step solution process:
  1. Build the solution operator. “Build stage.”
  2. Apply the solution operator. “Solve stage.”

  In many environments, both steps have $O(N)$ (or $O(N \log^k N)$) complexity.

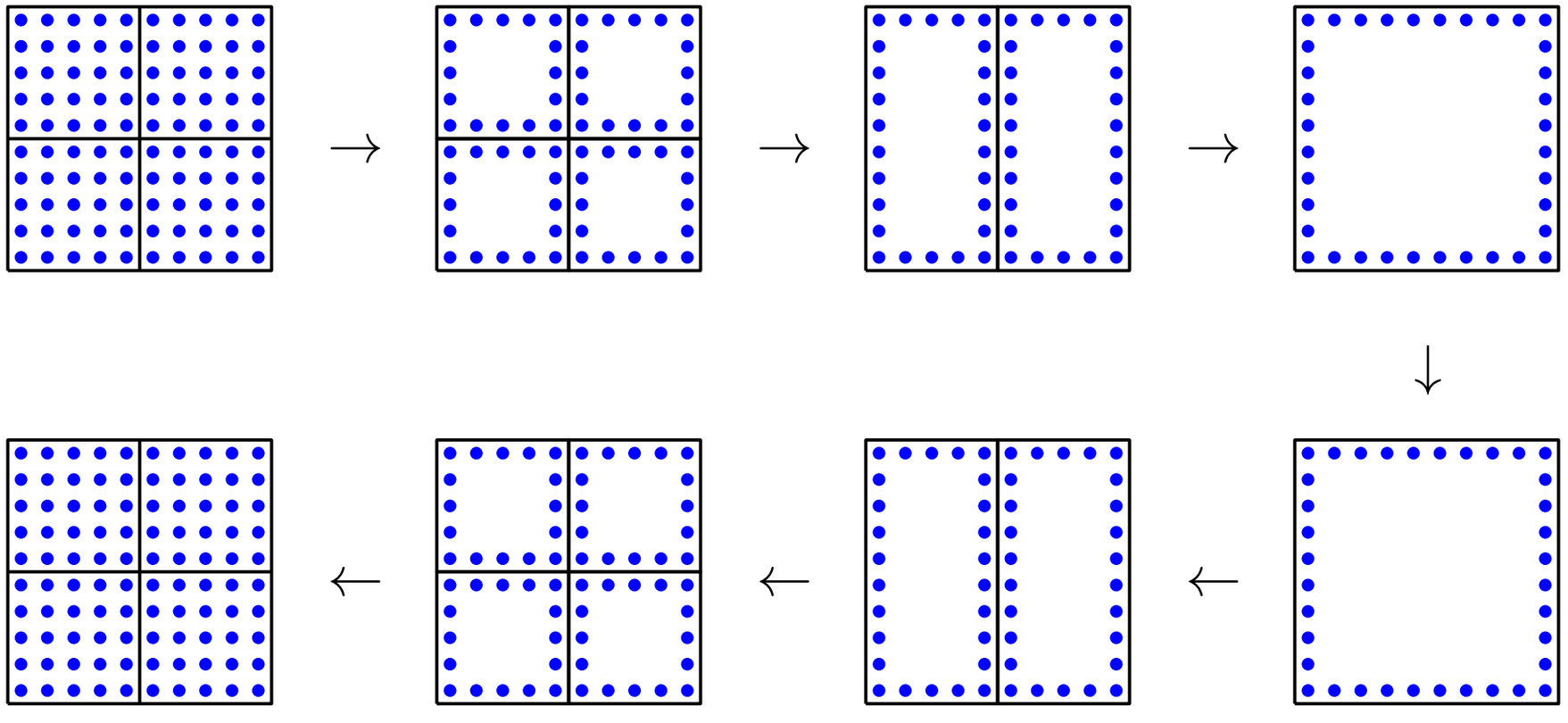
Recall that we are interested in solving the PDE $\begin{cases} A\,u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ B\,u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$ (BVP)

Explicit solution formula: $u(\boldsymbol{x}) = \int_\Omega G(\boldsymbol{x},\boldsymbol{y})\,g(\boldsymbol{y})\,d\boldsymbol{y} + \int_\Gamma F(\boldsymbol{x},\boldsymbol{y})\,f(\boldsymbol{y})\,dS(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Omega.$ (SLN)

Overview of direct solver:

- Split domain into “small” pieces.
- Build a “solution operator” for each small piece by brute force.
- Form a tree of successively larger pieces by pairwise merging.

  For each piece, form its “solution operator” by merging its children’s operators.
- Exploit that each “solution operator” is compressible (e.g. Calderón-Zygmund op.).



Advantages of direct solvers over iterative solvers:

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

   The "solve stage" is very fast, similar to fast Poisson solvers, FMM, etc.

2. Solving problems intractable to iterative methods:
   - Scattering problems near resonant frequencies.
   - Ill-conditioning due to geometry (elongated domains, percolation, etc).
   - Ill-conditioning due to corners, cusps, etc. (Whether avoidable or not.)
   - Finite element and finite difference discretizations.

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

3. Operator algebra:
   - "Gluing" together regions with different models/discretizations (e.g. "FEM-BEM coupling").
   - Eigenvalue problems: Vibrating structures, acoustics, band gap problems, ...
   - Operator multiplication, factorizations, etc.

Outline of talk

Part 1: Introduction (done).

Part 2: Randomized algorithms for accelerating matrix factorization algorithms.

Part 3: Direct solvers for sparse systems arising from the discretization of PDEs.

Part 4: Direct solvers for dense systems arising from the discretization of integral equations.

Part 2 (of 4): Randomized algorithms for accelerating matrix computations

Problem: Given an $m \times n$ matrix $\mathbf{A}$, and a target rank k, where $k \ll \min(m, n)$, we seek to compute an approximate partial singular value decomposition:

$$\begin{array}{ccccc} \mathbf{A} & \approx & \mathbf{U} & \mathbf{D} & \mathbf{V}^*, \\ m \times n & & m \times k & k \times k & k \times n \end{array}$$

with $\mathbf{U}$ and $\mathbf{V}$ having orthonormal columns, and $\mathbf{D}$ diagonal.

Solution: Pick an over-sampling parameter p, say $p = 5$. Then proceed as follows:

1. Draw an $n \times (k + p)$ Gaussian random matrix $\mathbf{R}$. `R = randn(n,k+p)`
2. Form the $m \times (k + p)$ sample matrix $\mathbf{Y} = \mathbf{A}\,\mathbf{R}$. `Y = A * R`
3. Form an $m \times (k + p)$ orthonormal matrix $\mathbf{Q}$ s. t. $\mathrm{ran}(\mathbf{Y}) = \mathrm{ran}(\mathbf{Q})$. `[Q, ~] = qr(Y)`
4. Form the $(k + p) \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$. `U = Q * Uhat`
7. Optional: Truncate the last p terms in the computed factors.

Joint work by P.G. Martinsson, V. Rokhlin, M. Tygert (2005/2006).

Input: An $m \times n$ matrix **A**, a target rank k, and an over-sampling parameter p (say $p = 5$).	
Output: Rank-$(k+p)$ factors **U**, **D**, and **V** in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix **R**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix **Q** s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

- Conceptually similar to a block Krylov method, but taking only a single step.

- A single step is enough due to mathematical properties of Gaussian matrices.
  We have shown (joint work with Joel Tropp of Caltech) that

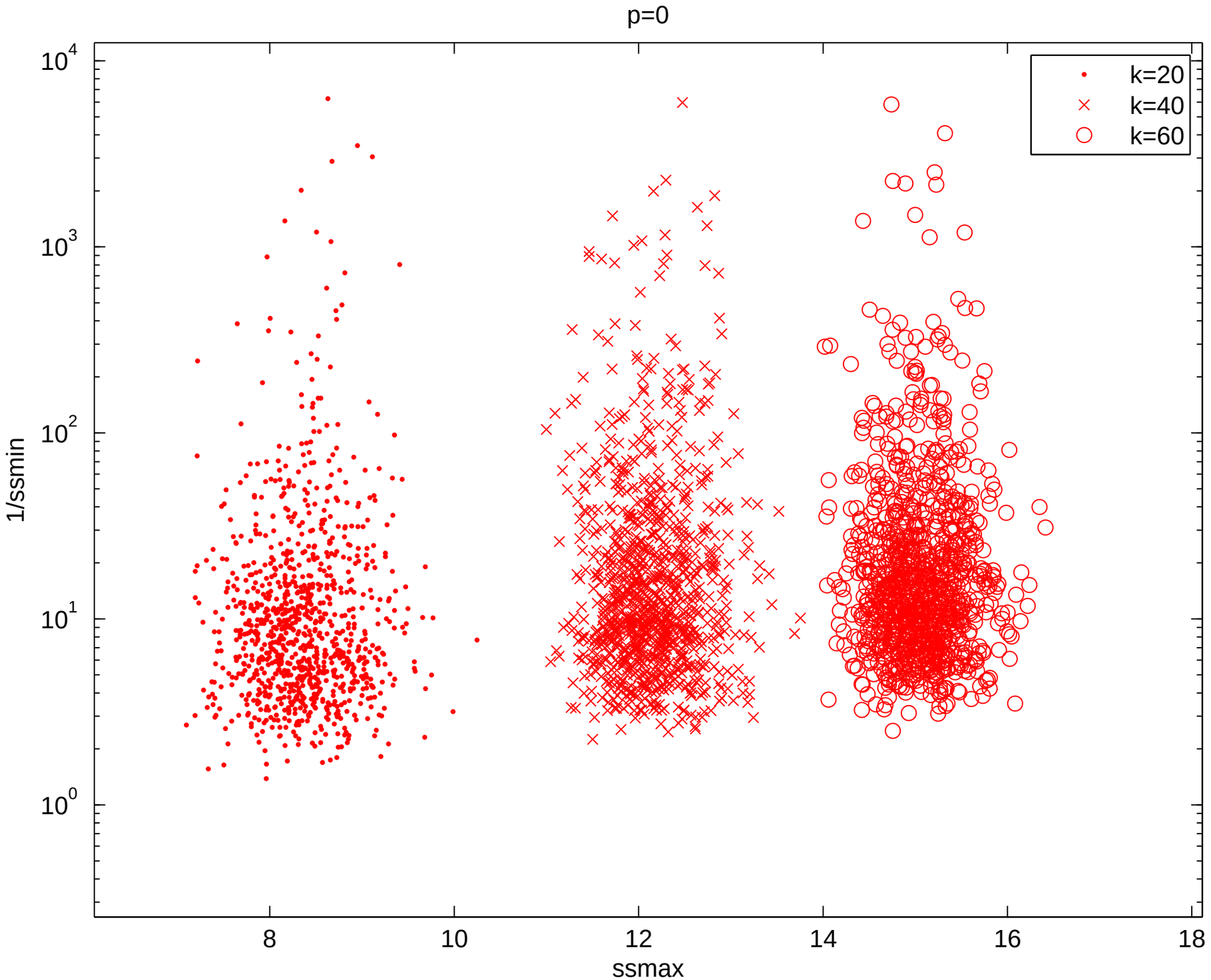
$$\mathbb{E}\big[\|\mathbf{A} - \mathbf{UDV}^*\|_{\text{Frob}}\big] \quad \sim \quad \frac{1}{\nu_{k,p}} \quad \times \quad \inf\{\|\mathbf{A} - \mathbf{B}\|_{\text{Frob}} : \ \mathbf{B} \text{ has rank } k\},$$

  where $\nu_{k,p}$ is the smallest singular value of a Gaussian matrix of size $k \times (k+p)$.

- For $p = 0$ (a square Gaussian matrix), it is well known that $\frac{1}{\nu_{k,p}}$ is huge and highly variable.

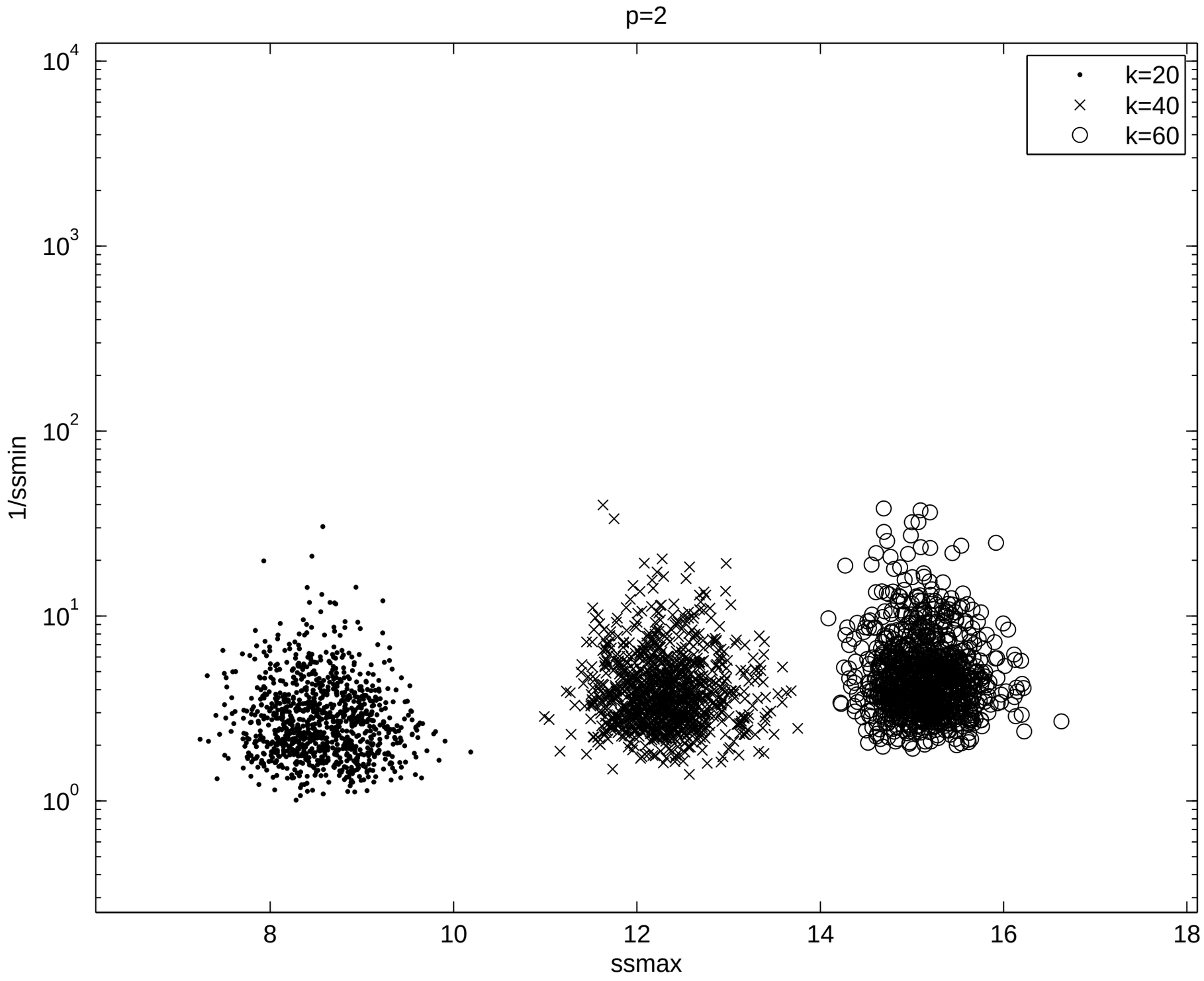
- Key fact [Chen and Dongarra, etc.]:
  As p increases, $\frac{1}{\nu_{k,p}}$ quickly becomes small and stable.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 0$



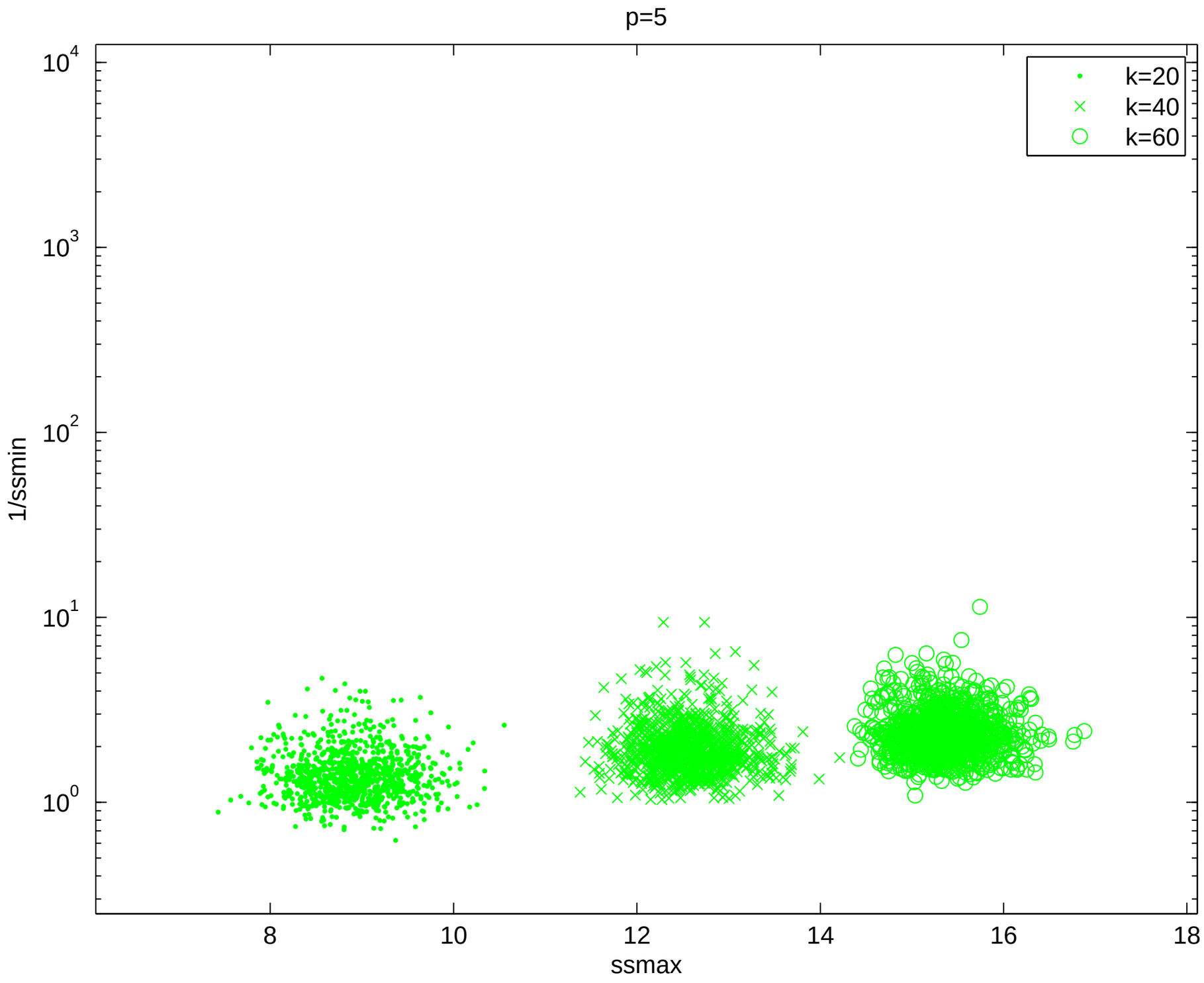
$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 2$



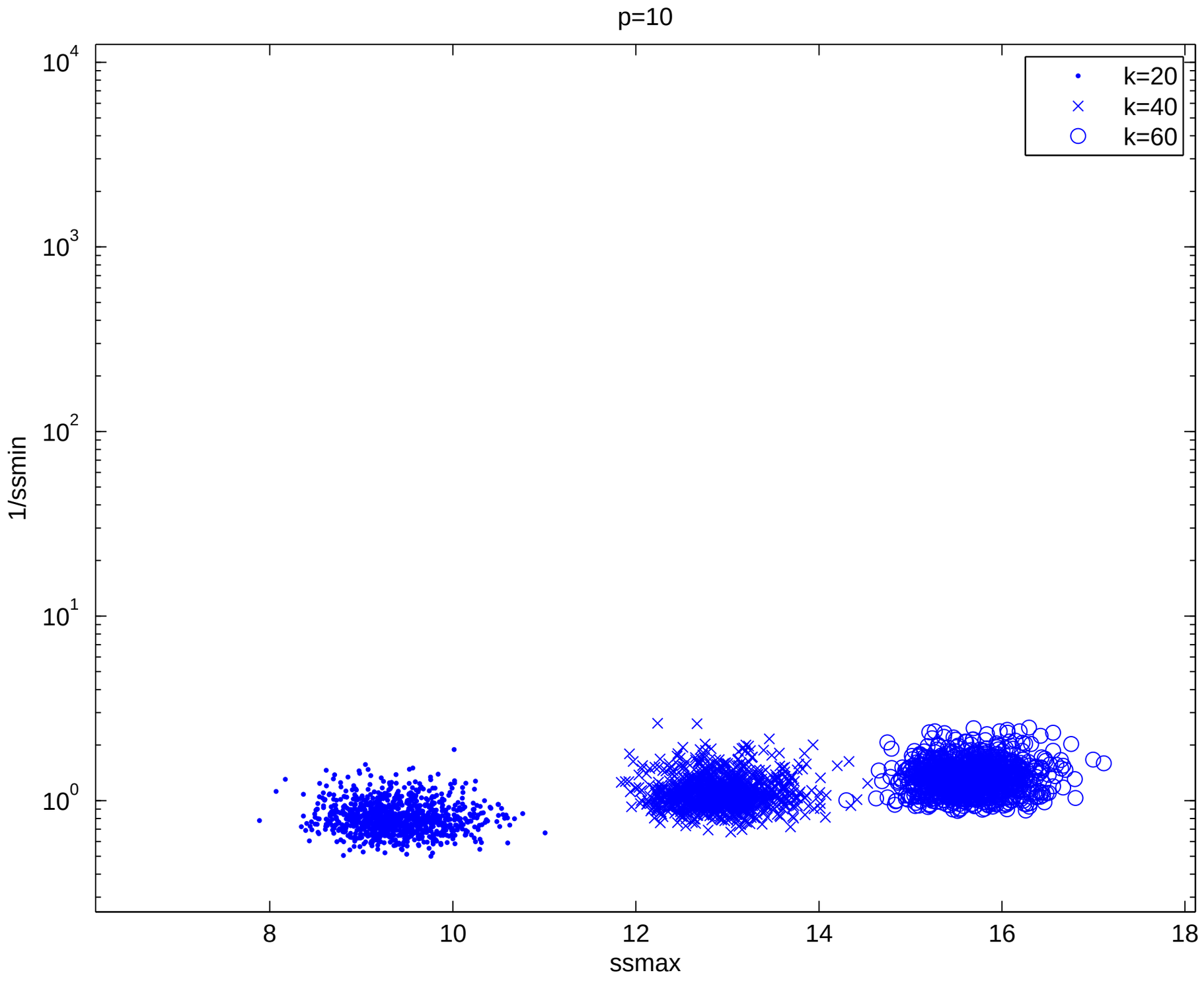
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 5$



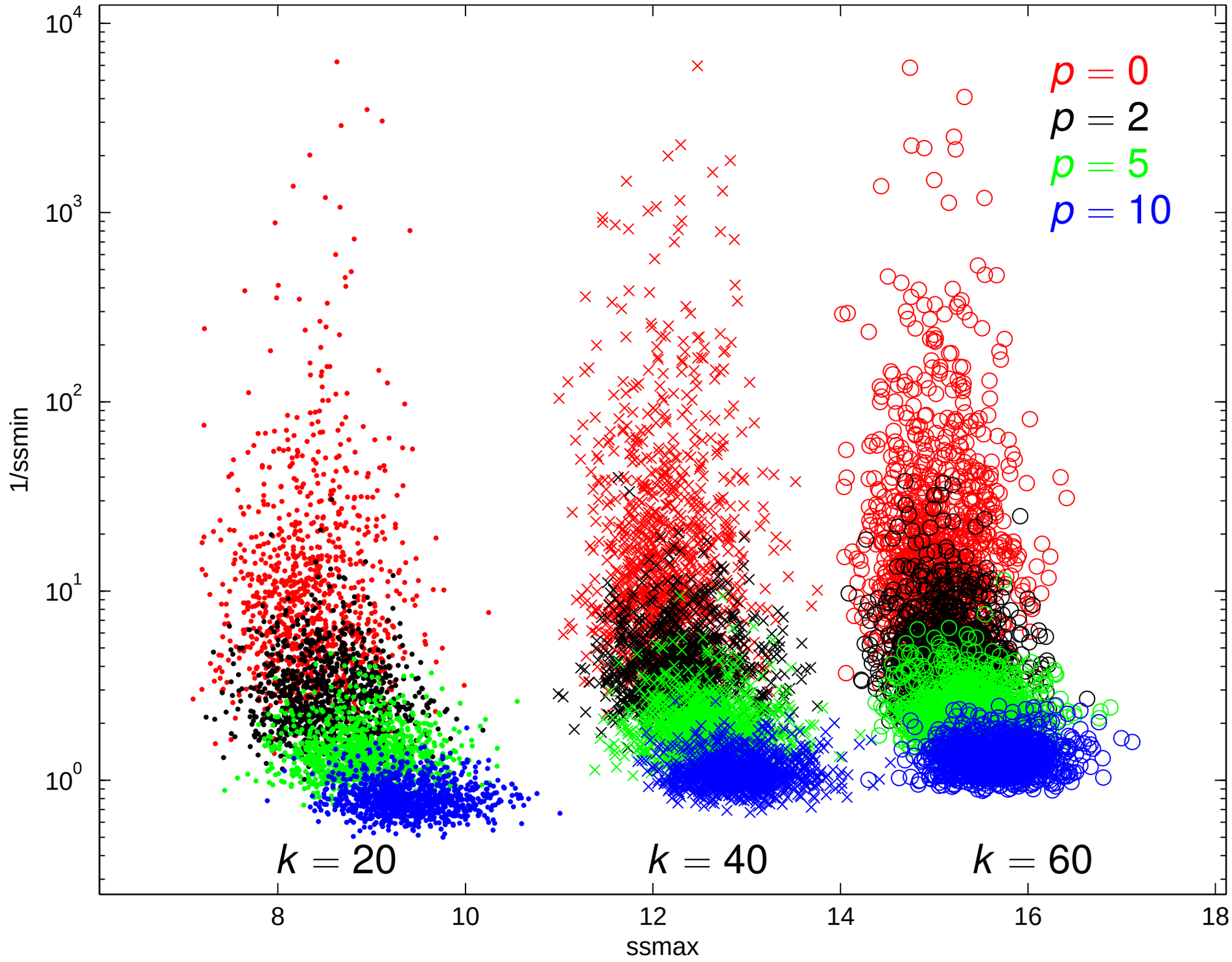
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 10$



$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

Scatter plot showing distribution of $k \times (k + p)$ Gaussian matrices.



$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

Bound on the expectation of the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{R}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AR})$.

If $p \geq 2$, then

$$\mathbb{E}\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|_{\text{Frob}} \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2},$$

and

$$\mathbb{E}\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}.$$

Ref: Halko, Martinsson, Tropp, 2009 & 2011

Large deviation bound for the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{R}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \mathtt{orth}(\mathbf{AR})$.

If $p \geq 4$, and u and t are such that $u \geq 1$ and $t \geq 1$, then

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + t\sqrt{\frac{3k}{p+1}} + u\,t\,\frac{e\sqrt{k+p}}{p+1}\right)\sigma_{k+1} + \frac{t\,e\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}$$

except with probability at most $2\,t^{-p} + e^{-u^2/2}$.

Ref: Halko, Martinsson, Tropp, 2009 & 2011; Martinsson, Rokhlin, Tygert (2006)

---

u and t parameterize "bad" events — large u, t is bad, but unlikely.

Certain choices of t and u lead to simpler results. For instance,

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + 16\sqrt{1 + \frac{k}{p+1}}\right)\sigma_{k+1} + 8\frac{\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability at most $3\,e^{-p}$.

Large deviation bound for the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{R}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \mathtt{orth}(\mathbf{AR})$.

If $p \geq 4$, and u and t are such that $u \geq 1$ and $t \geq 1$, then

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + t\sqrt{\frac{3k}{p+1}} + u\,t\,\frac{e\sqrt{k+p}}{p+1}\right)\sigma_{k+1} + \frac{t\,e\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}$$

except with probability at most $2\,t^{-p} + e^{-u^2/2}$.

Ref: Halko, Martinsson, Tropp, 2009 & 2011; Martinsson, Rokhlin, Tygert (2006)

---

u and t parameterize "bad" events — large u, t is bad, but unlikely.

Certain choices of t and u lead to simpler results. For instance,

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + 6\sqrt{(k+p)\cdot p\log p}\right)\sigma_{k+1} + 3\sqrt{k+p}\left(\sum_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability at most $3\,p^{-p}$.

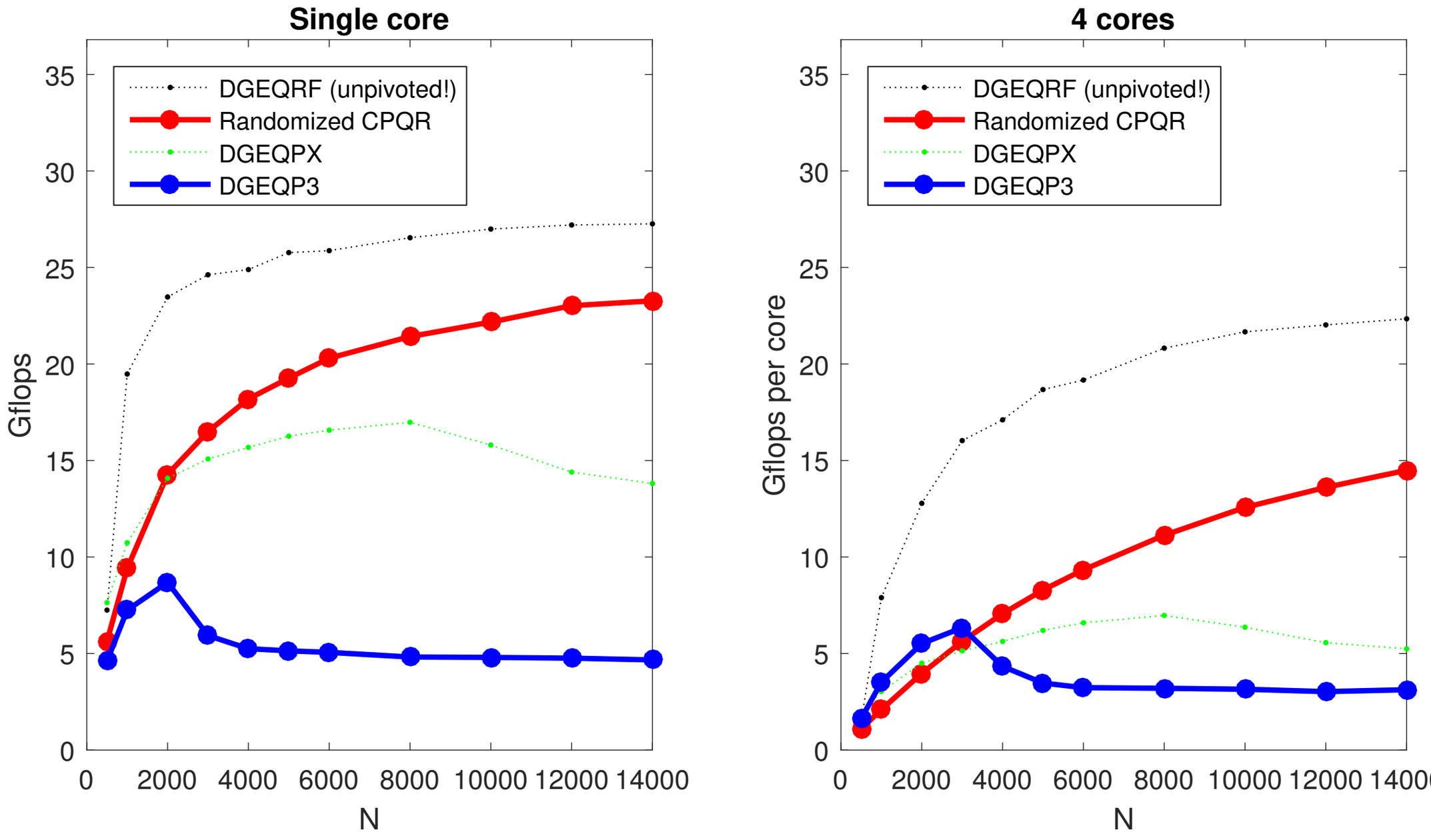
Input: An $m \times n$ matrix **A**, a target rank k, and an over-sampling parameter p (say $p = 5$).	
Output: Rank-$(k+p)$ factors **U**, **D**, and **V** in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix **R**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix **Q** s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

- It is simple to adapt the scheme to the situation where the *tolerance is given,* and the rank has to be determined adaptively.
- Vastly reduced *communication* compared to text book methods.
- The method has quickly become widely used for *data analysis* (huge datasets; data stored out-of-core; distributed memory machines; cloud computing; etc).
- Minor modifications lead to a *streaming algorithm* that never stores **A** at all.
- The flop count can be reduced from $O(mnk)$ to $O(mn\log k)$ by using a so called "fast Johnson-Lindenstrauss" transform. Speed gain of factor between 2 and 8 for matrices of size, e.g., 3000×3000.
- Accuracy of the basic scheme is good when the singular values decay reasonably fast. When they do not, the scheme can be combined with Krylov-type ideas: *Taking one or two steps of subspace iteration vastly improves the accuracy.*

Input: An $m \times n$ matrix **A**, a target rank k, and an over-sampling parameter p (say $p = 5$).	
Output: Rank-$(k+p)$ factors **U**, **D**, and **V** in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix **R**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix **Q** s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

The randomized algorithm is particularly powerful in strongly communication constrained environments (huge matrices stored out-of-core, distributed memory parallel computers, GPUs), but it is also very beneficial in a "classical" computing environment (matrix stored in RAM, standard CPU, etc).

Very recent result: Can solve the long-standing problem of how to *block* the column pivoted QR factorization!

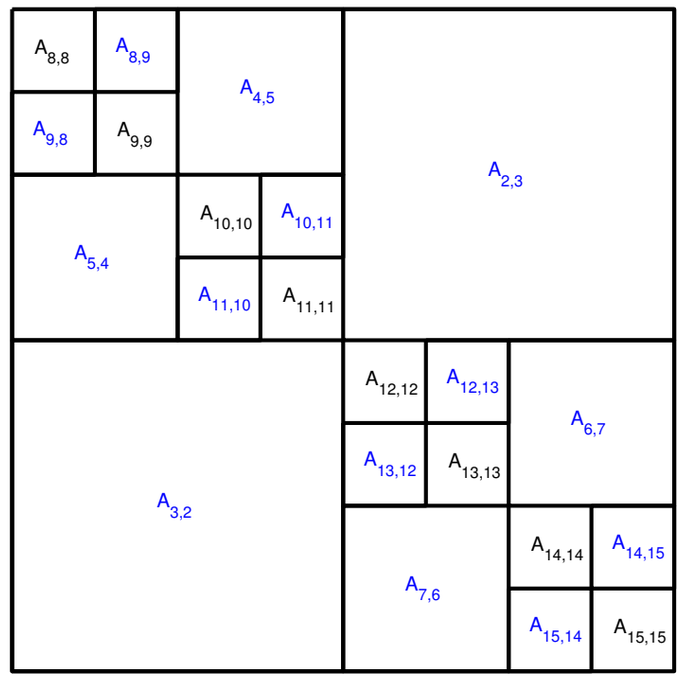


Speedup attained by randomized methods for computing a full column pivoted QR factorization of an $N \times N$ matrix. The thick blue line shows the speed of LAPACK (DGEQP3), and the thick red line the randomized method. We also include the speed of LAPACK's unpivoted QR factorization (black) and a competing "panel pivoting" scheme (green). We use Release 3.4.0 of LAPACK and linked it to the Intel MKL library Version 11.2.3. The top of the graphs indicate the theoretical maximal flop rate for the Intel Xeon E5-2695 CPU of 36.8Gflops (turbo boost was turned off). Joint work with G. Quintana-Ortí, N. Heavner, and R. van de Geijn.

Randomized approximation of "rank-structured" matrices

Recall that a *rank-structured matrix* is a matrix whose off-diagonal blocks have low numerical rank. Many "formats" have been proposed, including: (1) "Fast Multipole Method" matrices. Panel clustering. (2) $\mathcal{H}$- and $\mathcal{H}^2$-matrices. (3) Hierarchically Block Separable (HBS) matrices, a.k.a. "HSS" matrices. (4) S-matrices, a.k.a. HODLR matrices.

All these formats allow for (more or less) efficient matrix computations such as matrix-vector multiply, matrix-matrix multiply, LU factorization, matrix inversion, forming of Schur complements, etc.



Randomized matrix compression is extremely useful here! We have developed *black-box* compression algorithms that construct a data-sparse representation of a rank-structured matrix **A** that rely only on being able to apply **A** to a small number of random vectors.

Vast numbers of blocks are compressed jointly and in parallel.

- P.G. Martinsson, *A fast randomized algorithm for computing a Hierarchically Semi-Separable representation of a matrix.* 2008 arxiv report. 2011 SIMAX paper.
- Later improvements by Jianlin Xia, Sherry Li, etc.
- L. Lin, J. Lu, L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, JCP 2011.
- P.G. Martinsson, "Compressing rank-structured matrices via randomized sampling." Arxiv.org report #1503.07152. In review.

Randomized algorithms for matrix computations — future directions:

- *Exploit communication efficiency.* Several different environments:
  - Classical single and multicore architectures. We have already demonstrated dramatic improvements, much more can be done.
  - Develop libraries for distributed memory low-rank approximation: partial SVD, PCA, LSI, finding spanning rows and columns, etc.
  - GPU and massively multicore environments.
  - Mobile computing — exploit energy efficiency.
  - "Big data" — cloud computing, machine learning, computational statistics, etc.

- Better algorithms for *rank-structured matrices.* Growing demand for structured matrix codes — human cost of software development currently a bottleneck.

- Optimize for matrices from particular applications. Sparse, in particular.

- Further improvements in *theoretical understanding.* Open questions concerning randomized pivoting, *structured* random matrices (randomized DFT / Hadamard / wavelet transforms/ ...), connections to compressive sensing, etc.

- Develop better rank-revealing QR factorizations.
  (Much better! Computational profile of CPQR with accuracy of SVD.)

Part 3 (of 4): Direct solvers for discretized PDEs

- Consider an elliptic PDE

  (BVP)
  $$\begin{cases} Au(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ Bu(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

  where Ω is a domain in $\mathbb{R}^2$ or $\mathbb{R}^3$ with boundary Γ.

  Think Laplace, Helmholtz, Yukawa, time-harmonic Maxwell, etc.

- Discretize (BVP) using FEM / FD / ... to obtain a linear system

  $$\mathbf{Au} = \mathbf{b}.$$

- Given a computational tolerance ε, we now seek a *direct* (that is, *non-iterative*) algorithm that builds a dense matrix $\mathbf{S}$ such that

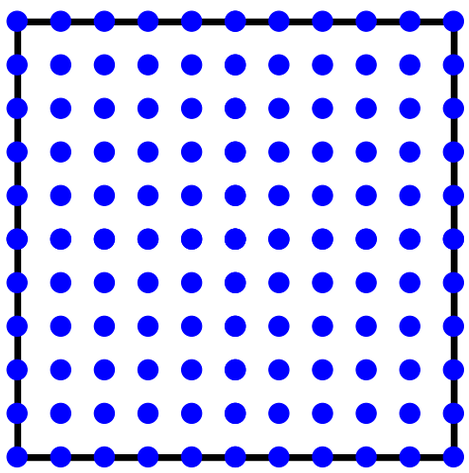
  $$\|\mathbf{S} - \mathbf{A}^{-1}\| \leq \varepsilon.$$

  The matrix $\mathbf{S}$ is represented in a *data-sparse* format.

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.



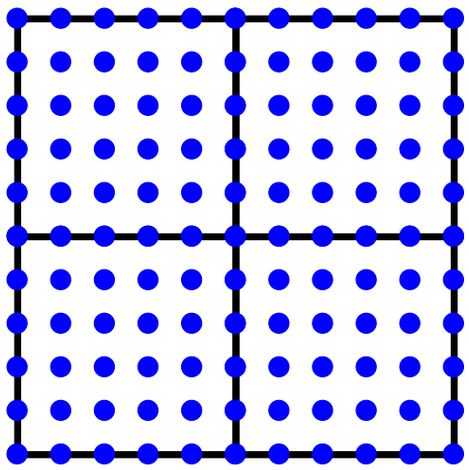
The original grid.

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into “small” patches we call “leaves” (they will be organized in a tree).



The original grid.

Outline of direct solver

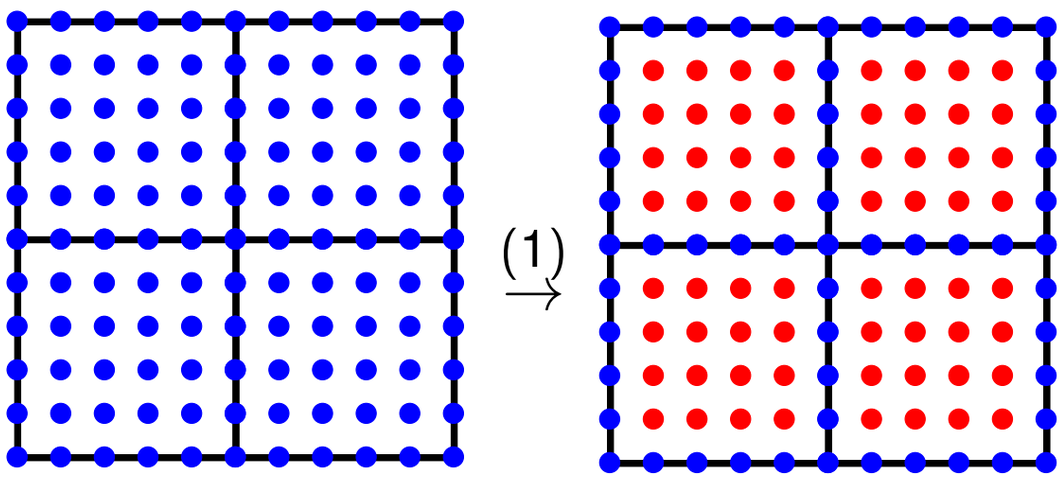
All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")



The original grid. *Leaves reduced.*

Outline of direct solver

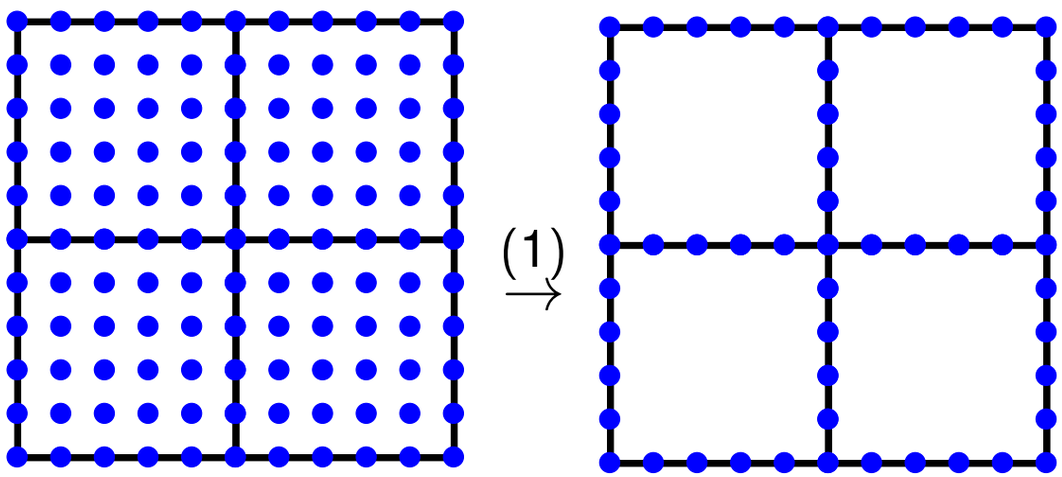
All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")



The original grid. *Leaves reduced.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

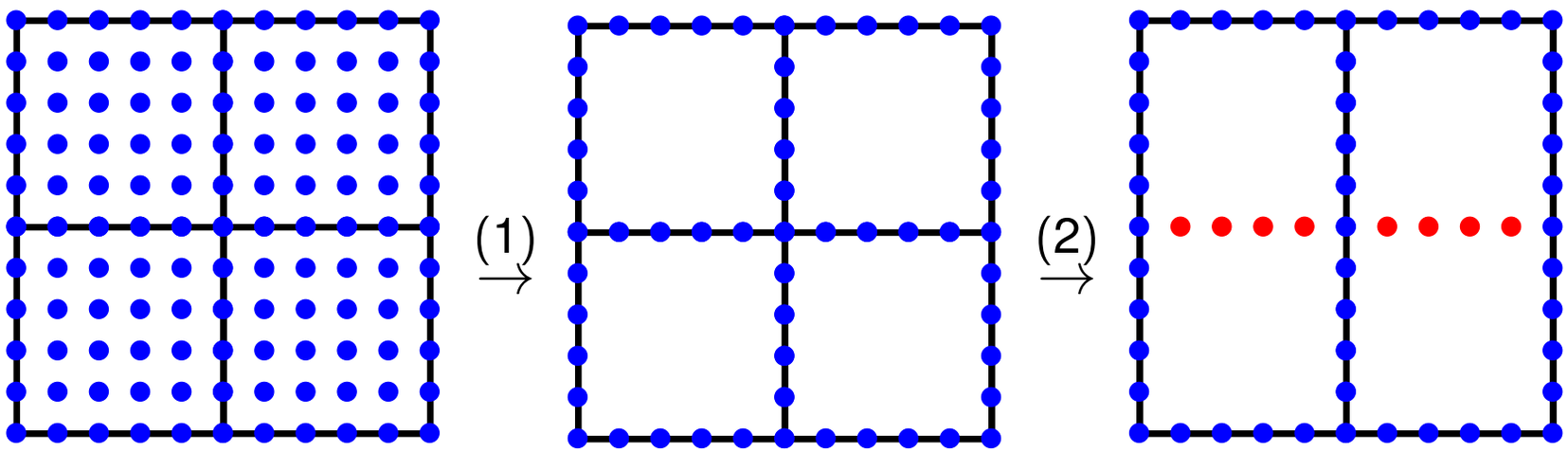
Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two.



The original grid. *Leaves reduced.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

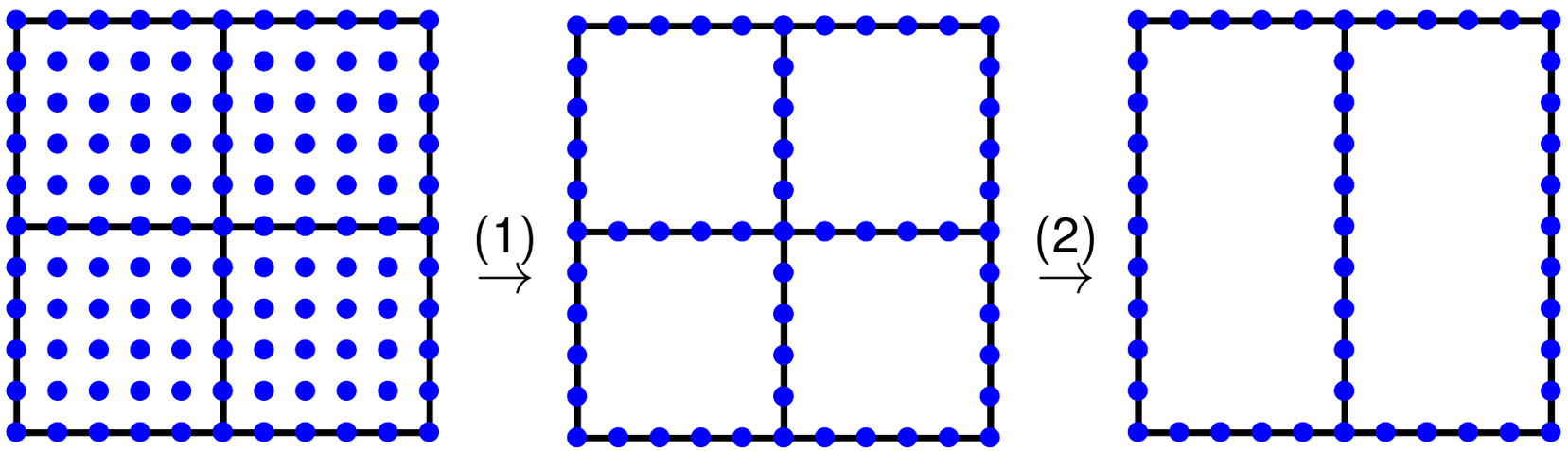
Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.



The original grid. *Leaves reduced.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

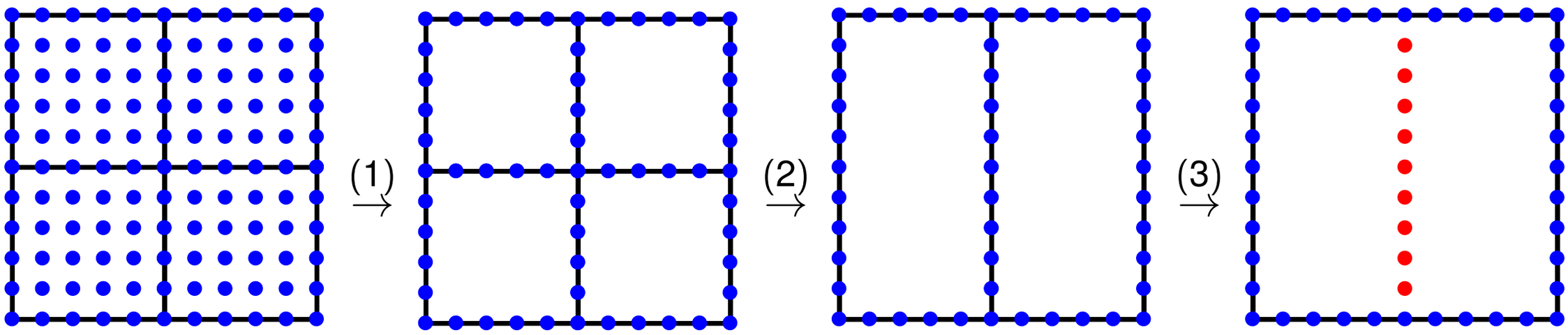
Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

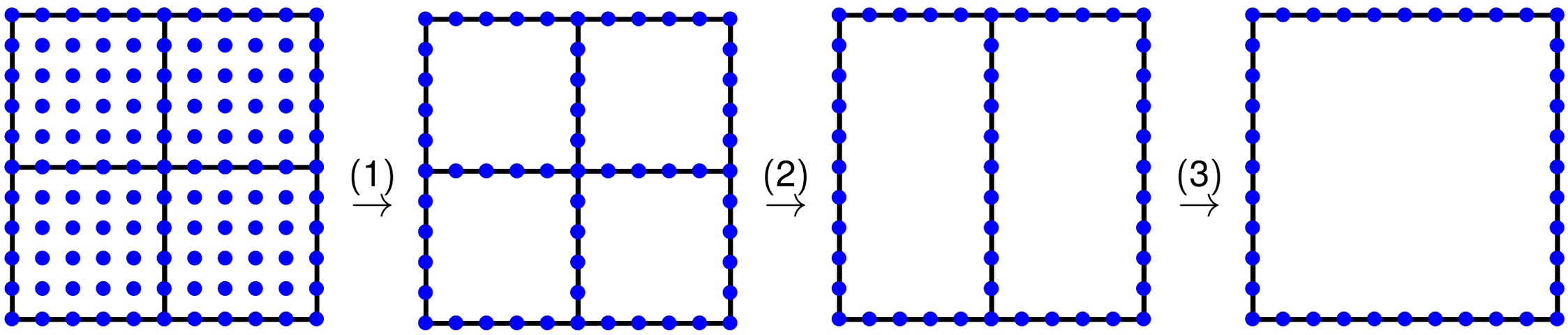
Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

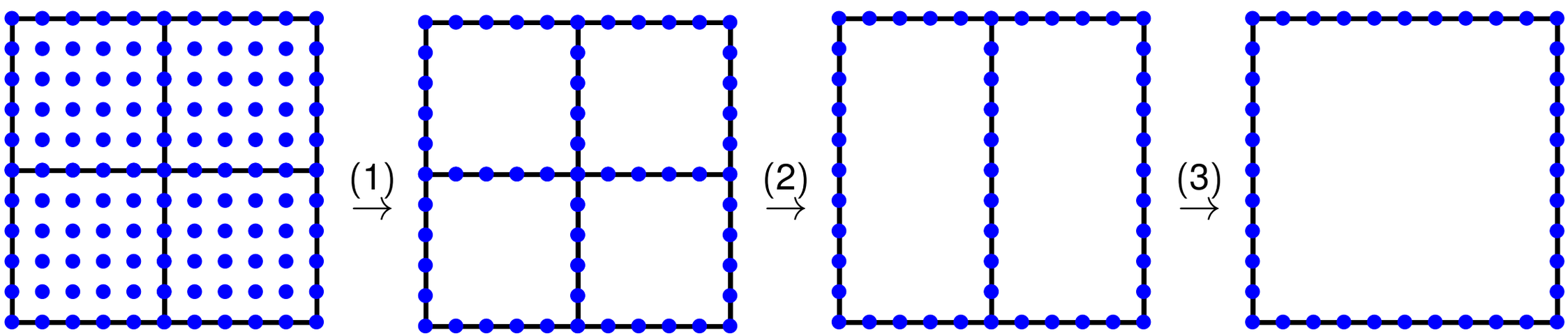
On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.

When you reach the top level, perform a solve on the reduced problem by brute force.



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Outline of direct solver

All direct solvers to be described are based on hierarchical domain decomposition.

Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

Split the domain into "small" patches we call "leaves" (they will be organized in a tree).

On each leaf, compute by "brute force" a local solution operator (e.g. a DtN operator).

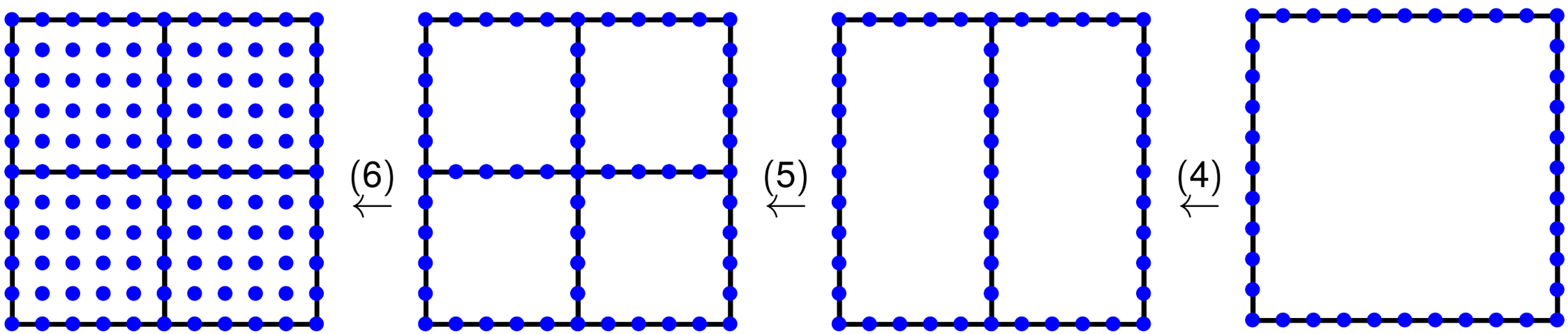
This eliminates "internal" grid points from the computation. ("Static condensation.")

Merge the leaves in pairs of two. For each pair, compute a local solution operator by combining the solution operators of the two leaves.

Continue merging by pairs, organizing the domain in a tree of patches.

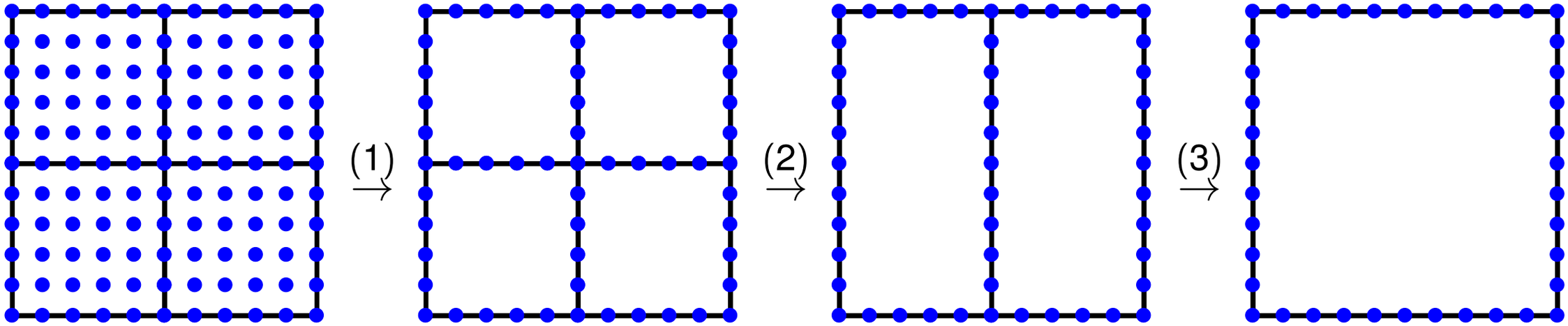
When you reach the top level, perform a solve on the reduced problem by brute force.

Then reconstruct the solution at all internal points via a downwards pass.



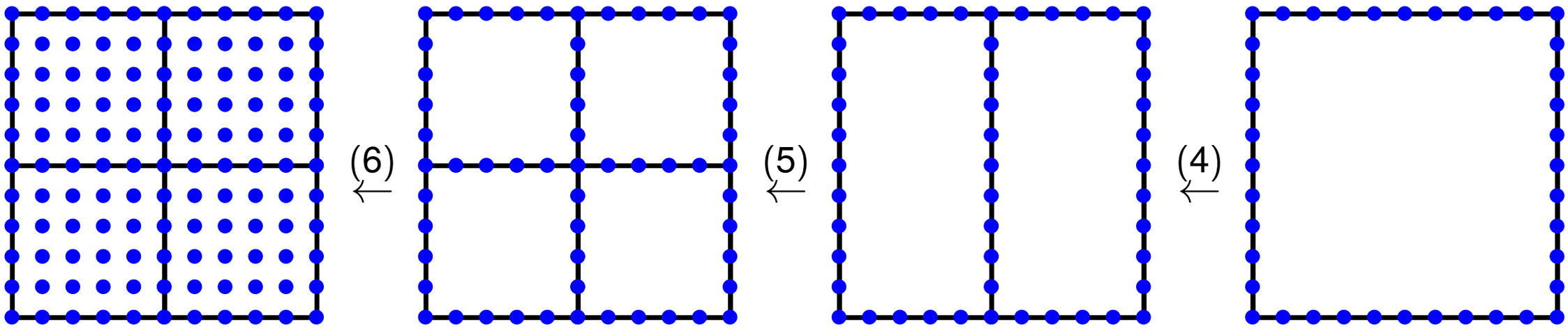
Full solution. *Solve.* *Solve.* *Top level solve.*

Upwards pass — build all solution operators:



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Downwards pass — solve for a particular data function (*very* fast!):



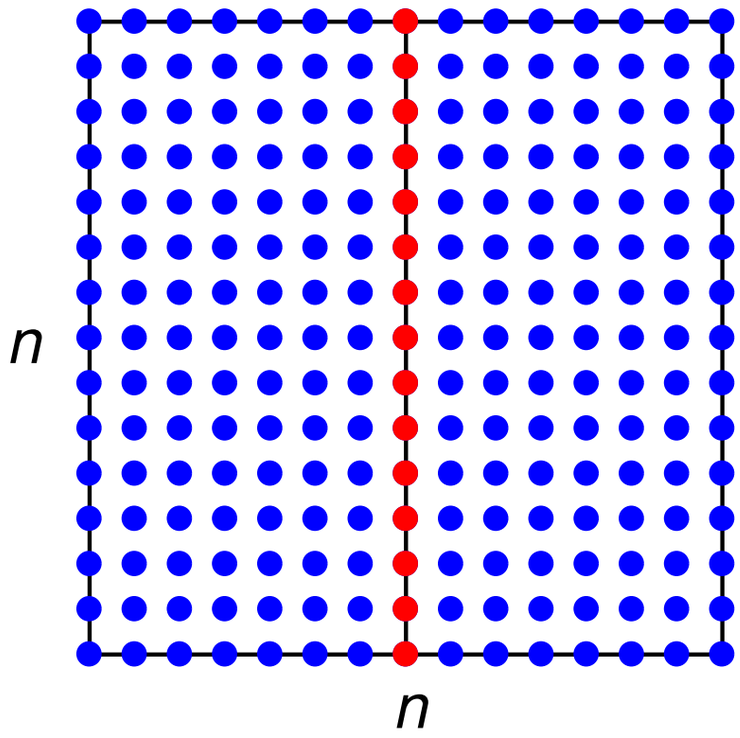
Full solution. *Solve.* *Solve.* *Top level solve.*

Note: *The computational template outlined is the same as the classical multifrontal / nested dissection method (George 1973, later I. Duff, T. Davis, etc).*

The direct solver described works very well for moderate problem sizes.

But problems arise as the number of discretization points increases ...

Consider a regular grid in 2D with $N = n \times n$ total nodes. The top level merge requires inversion of a matrix representing interactions between the red nodes:



$$N = n \times n$$

$$n = N^{1/2}$$

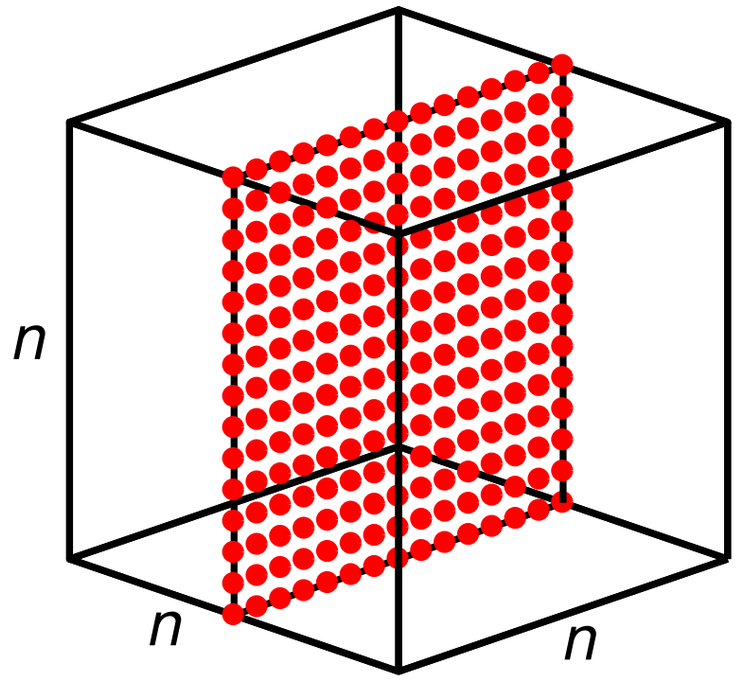
Since this dense matrix is of size $n \times n$, the cost for the merge is

$$\text{COST} \sim n^3 \sim (N^{1/2})^3 \sim N^{3/2}.$$

Problem: *3D is even worse!*

The direct solver described works very well for moderate problem sizes.
But problems arise as the number of discretization points increases ...

Consider a regular grid in 3D with $N = n \times n \times n$ total nodes. The top level merge requires inversion of a matrix representing interactions between the red nodes:



$$N = n \times n \times n$$
$$n = N^{1/3}$$

Since this dense matrix is of size $n^2 \times n^2$, the cost for the merge is

$$\text{COST} \sim n^6 \sim (N^{1/3})^6 \sim N^2.$$

Assertion: *The dense matrix very often behaves like a discretizated integral operator. It is rank-structured, and is amenable to "fast" matrix algebra.*

Exploiting the assertion on the previous page, we have in the last 10 years managed to reduce the asymptotic complexity of direct solvers for elliptic PDEs dramatically:

	Build stage		*Solve stage*	
2D	$N^{3/2}$	$\rightarrow N$	$N \log N$	$\rightarrow N$
3D	N^2	$\rightarrow N$	$N^{4/3}$	$\rightarrow N$

Key idea: Represent dense matrices using rank-structured formats (such as $\mathcal{H}$-matrices).

Nested dissection solvers with $O(N)$ complexity — *Le Borne, Grasedyck, & Kriemann (2007)*, *Martinsson (2009)*, *Xia, Chandrasekaran, Gu, & Li (2009)*, *Gillman & Martinsson (2011)*, *Schmitz & Ying (2012)*, *Darve & Ambikasaran (2013)*, *Ho & Ying (2015)*, etc.

$O(N)$ direct solvers for integral equations were developed by *Martinsson & Rokhlin (2005)*, *Greengard, Gueyffier, Martinsson, & Rokhlin (2009)*, *Gillman, Young, & Martinsson (2012)*, *Ho & Greengard (2012)*, *Ho & Ying (2015)*. Our direct solver is also related to an $O(N^{1.5})$ complexity direct solver for Lippman-Schwinger, see *Chen (2002) & (2013)*. Also related to direct solvers for integral equations based on $\mathcal{H}$ and $\mathcal{H}^2$ matrix arithemetic by Hackbusch (1998 and forwards), Börm, Bebendorf, etc.

Note: Complexity is not $O(N)$ if the nr. of "points-per-wavelength" is fixed as $N \rightarrow \infty$. This limits direct solvers to problems of size a couple hundreds of wave-lengths or so.

Claim: Direct solvers are ideal for combining with *high order discretization.*

- High order methods sometimes lead to more ill-conditioned systems.

  $\rightarrow$ *Can be hard to get iterative solvers to converge.*

- Direct solvers use a lot of memory per degree of freedom.

  $\rightarrow$ *You want to maximize the oomph per DOF.*

- Direct solvers are particularly well suited for "high" frequency wave problems.

  $\rightarrow$ *Need high order due to ill-conditioned physics.*

Problem: If you combine "nested dissection" with traditional discretization techniques (FD, FEM, etc), then the performance *plummets* as the order is increased.

Solution: Derive a new (or at least newish) discretization scheme that is directly tailored to work with fast direct solvers.

The Hierarchical Poincaré-Steklov Method

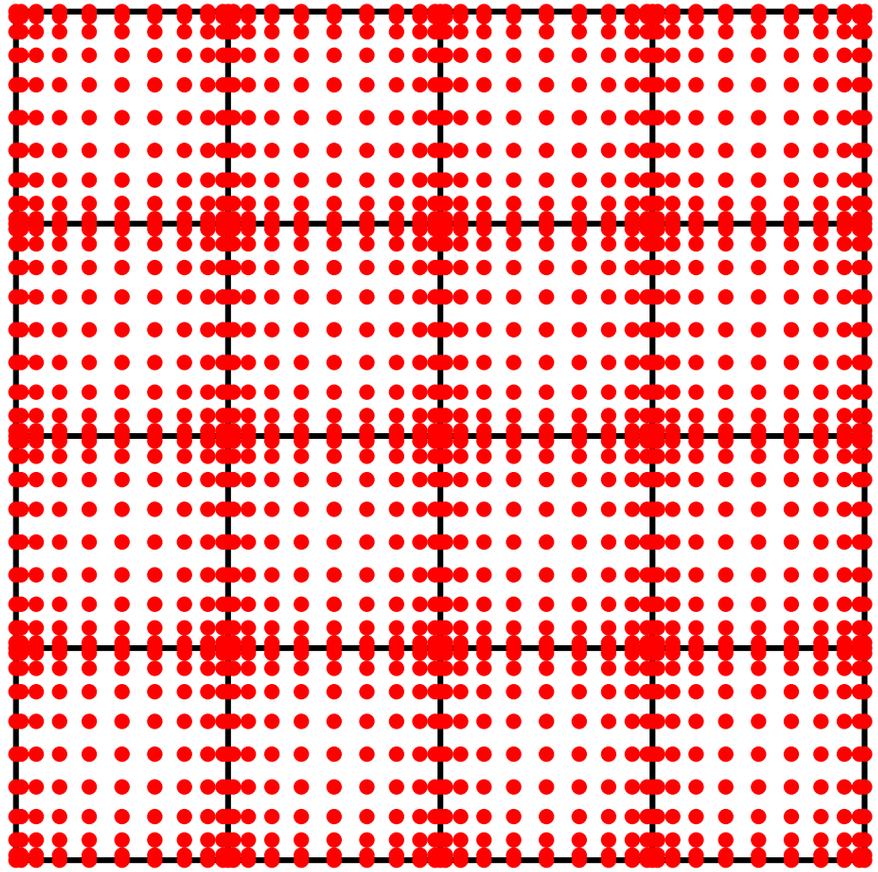
A direct solver based on a multidomain spectral collocation discretization

For simplicity, let us consider a "variable wave speed" Helmholtz problem in 2D: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$.

We assume u is smooth.



The unknown function u is represented as a vector holding approximations to its point-wise values at the grid points (collocation). Across domain boundaries, we enforce continuity of potentials and normal derivatives.

A global solution operator will be built using a nested-dissection type solver.

Prior work: *The discretization scheme is similar to existing composite (or "multi-domain") spectral collocation methods by Hesthaven and others. In particular: Pfeiffer, Kidder, Scheel, Teukolsky, (2003).*

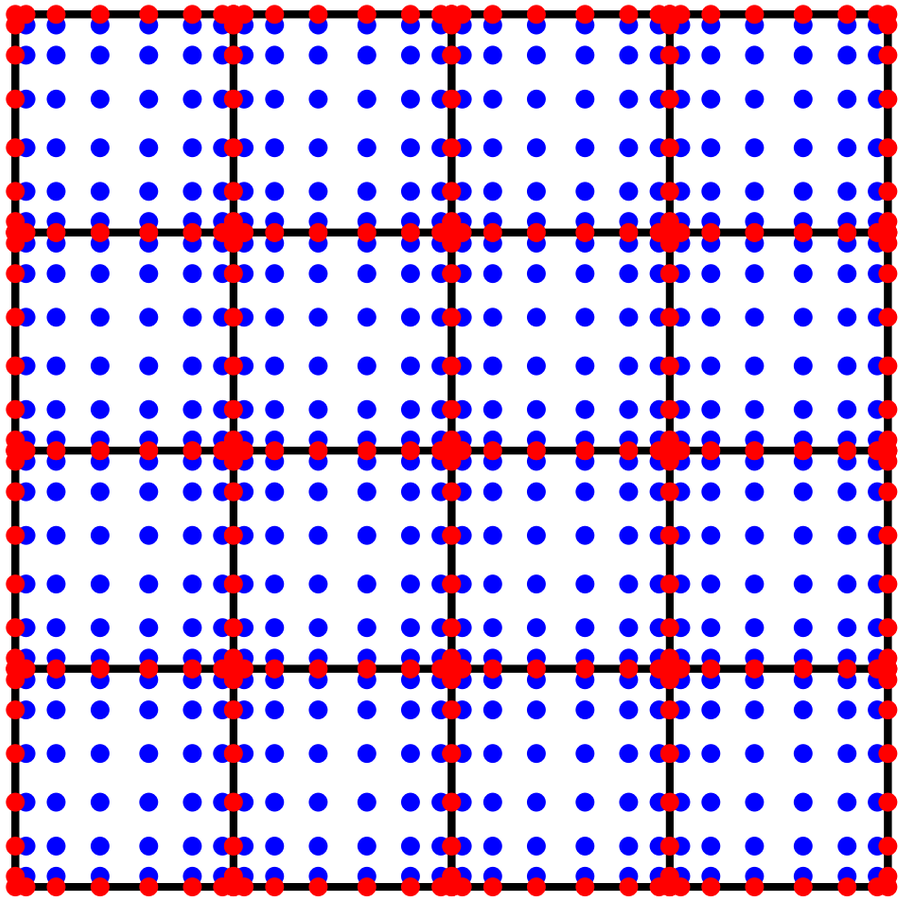
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Process leaves: Eliminate the interior (blue) nodes. ("Static condensation.")
Technically, we compute the Dirichlet-to-Neumann operator via a local spectral computation.



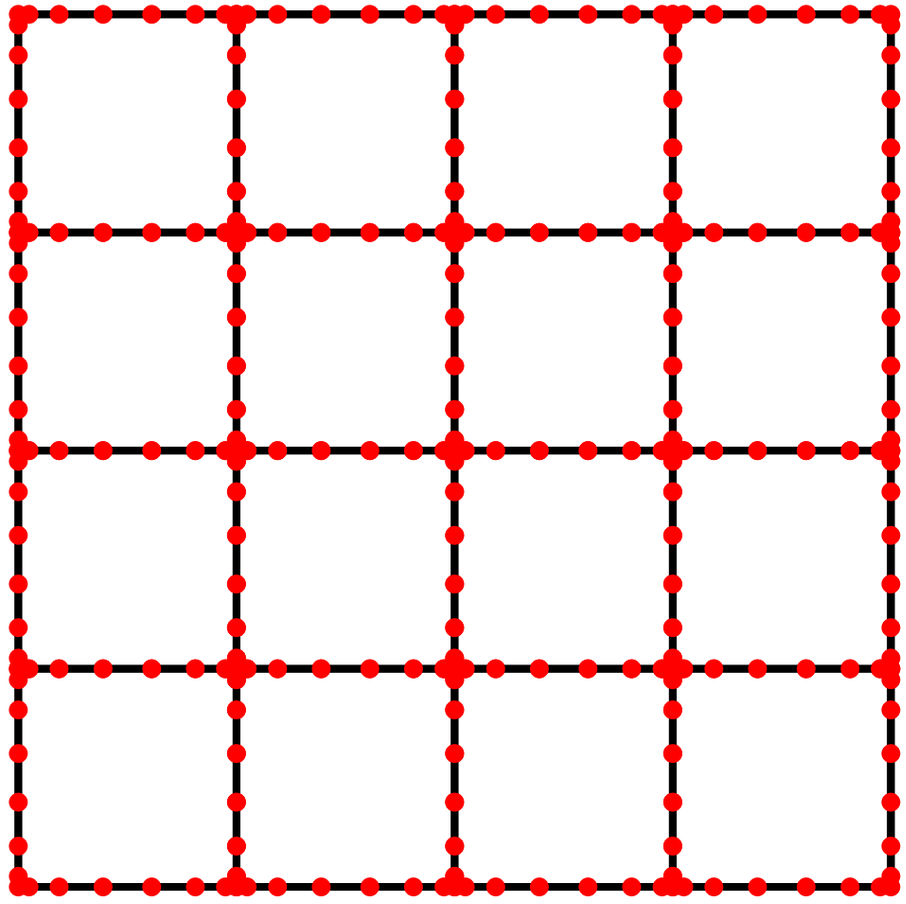
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Process leaves: Eliminate the interior (blue) nodes. ("Static condensation.")
Technically, we compute the Dirichlet-to-Neumann operator via a local spectral computation.



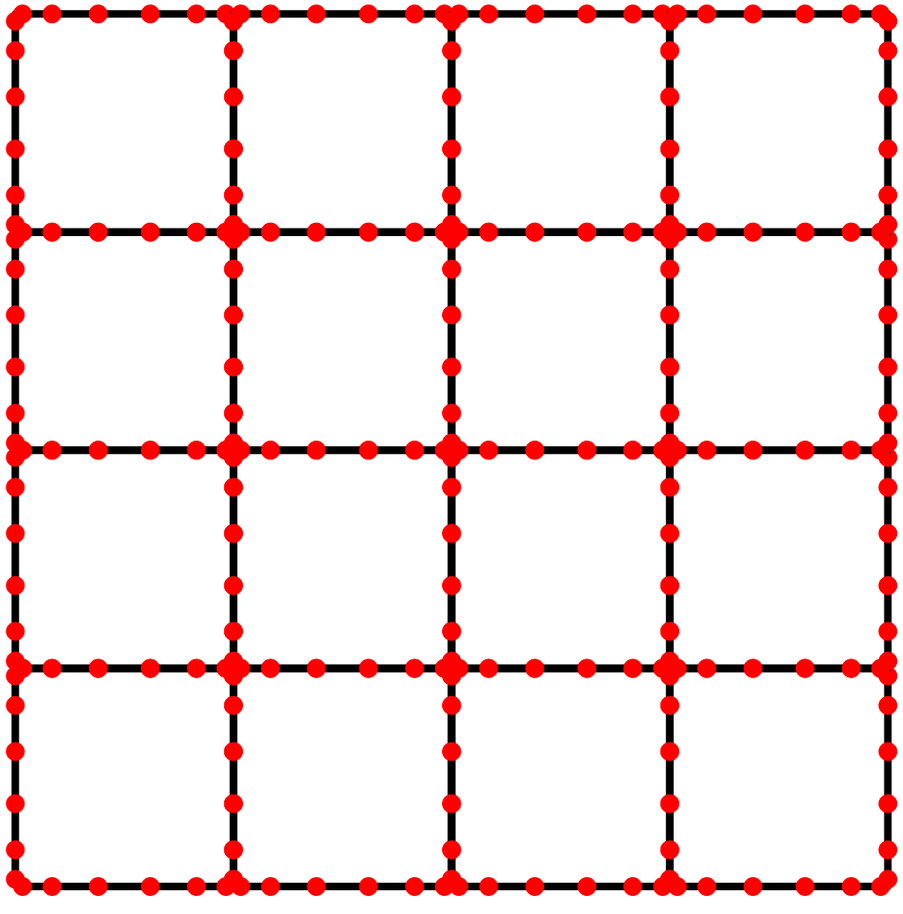
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Process leaves: Retabulate from Chebyshev to *Legendre nodes* on boundaries.



The Hierarchical Poincaré-Steklov Method

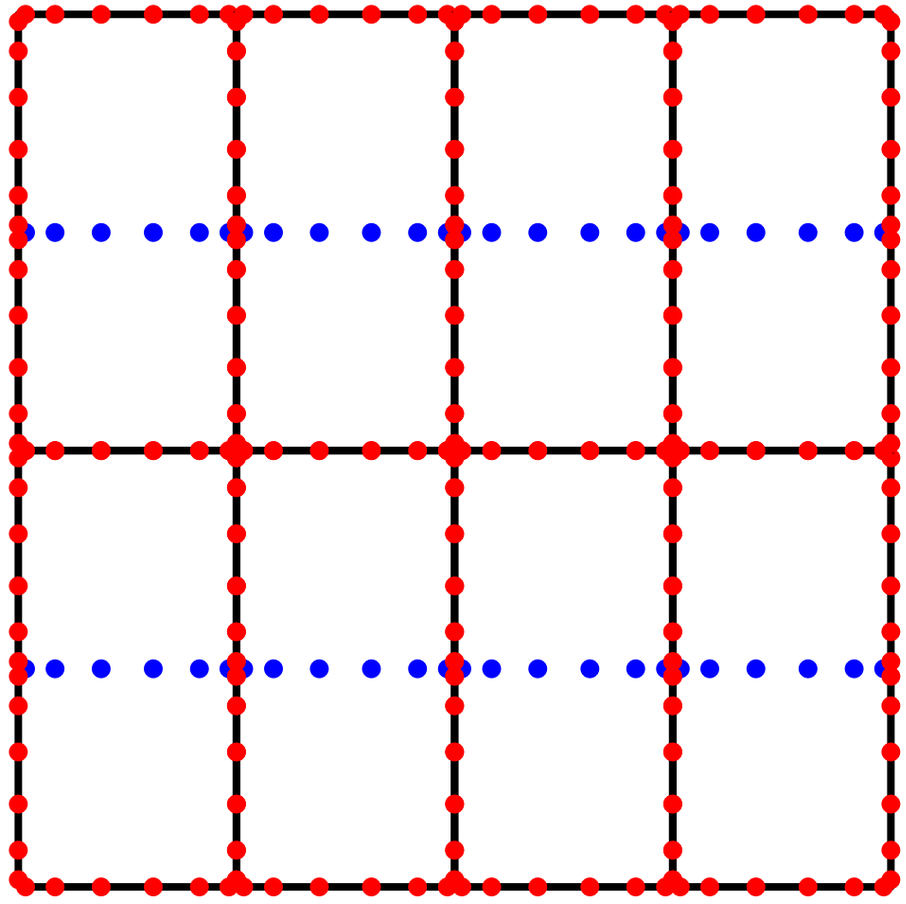
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

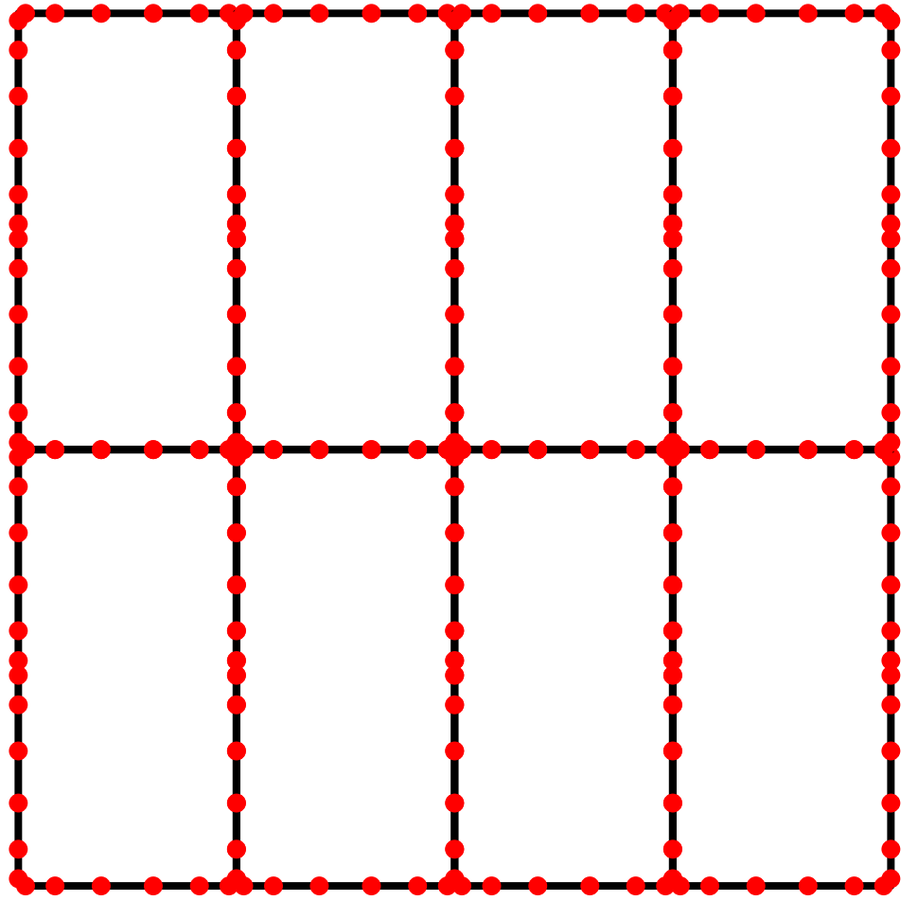
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

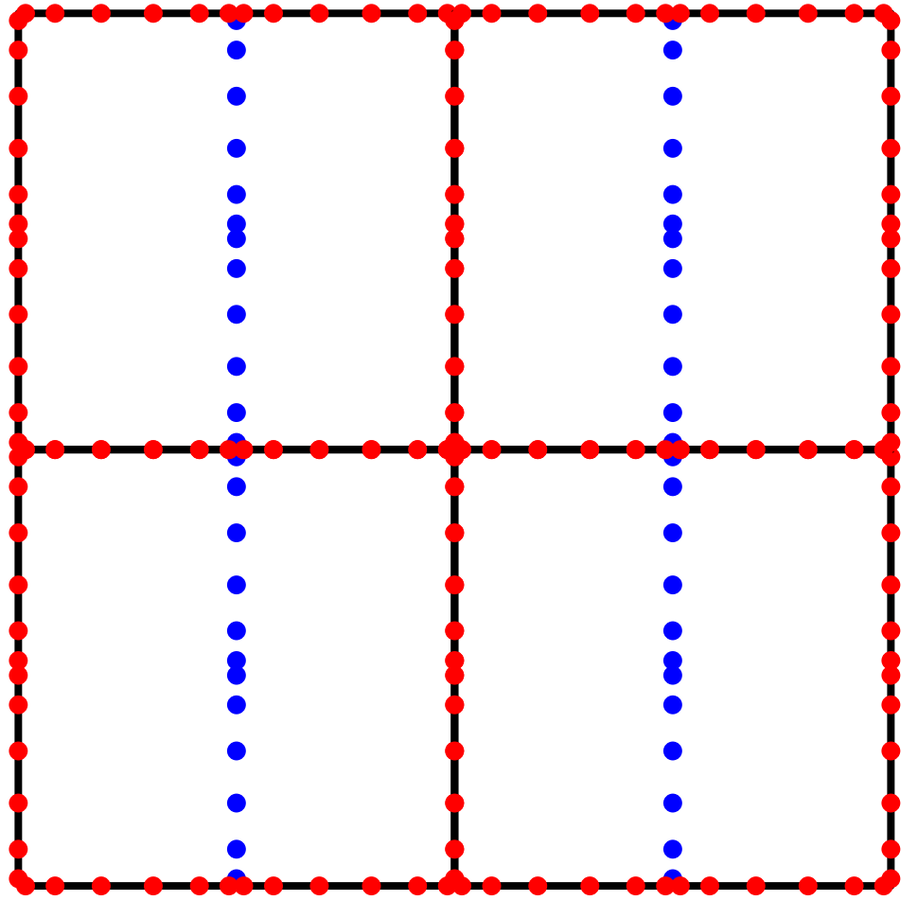
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

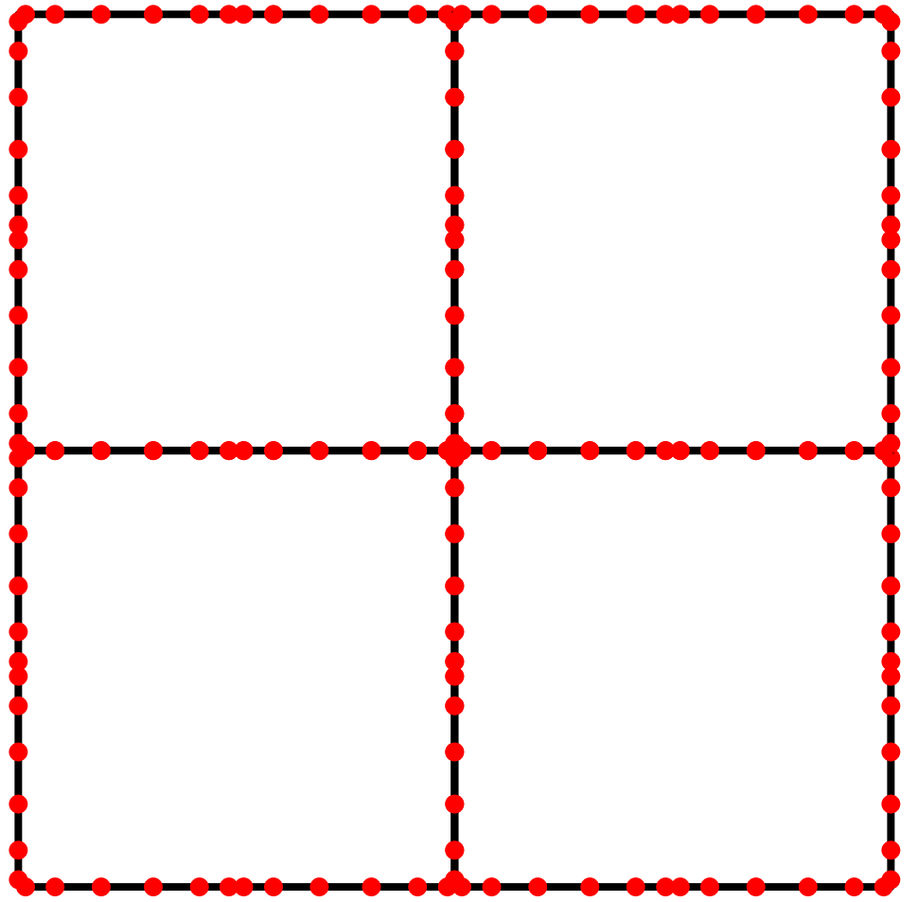
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

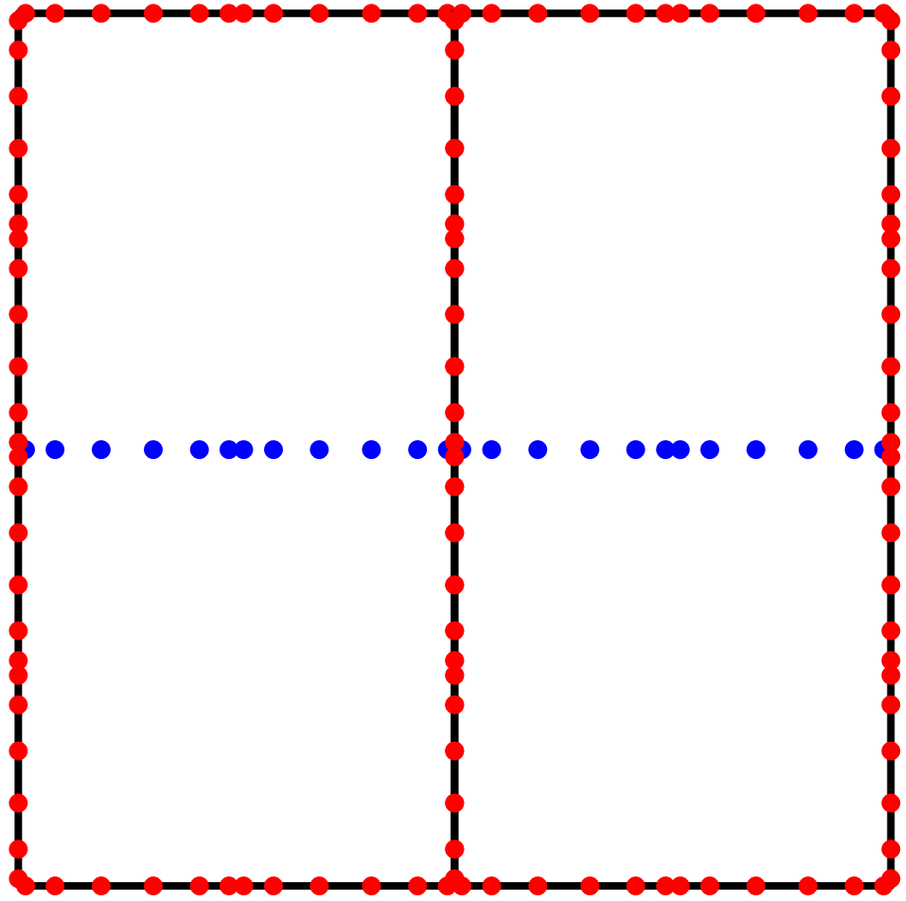
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

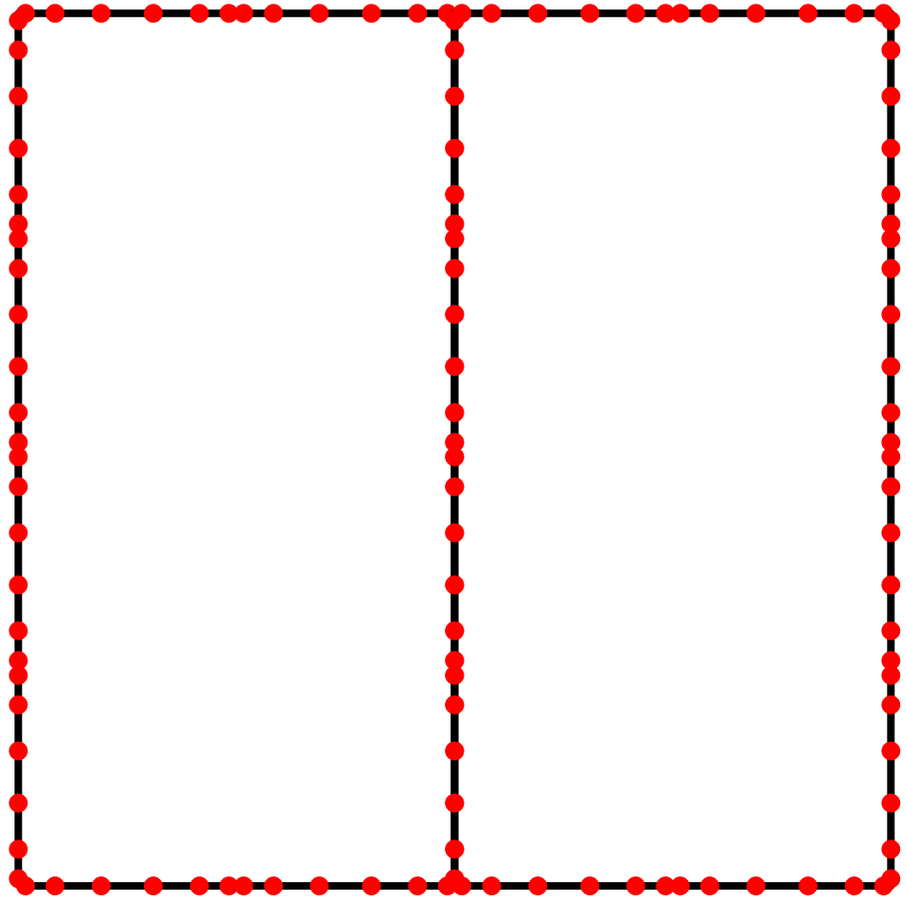
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

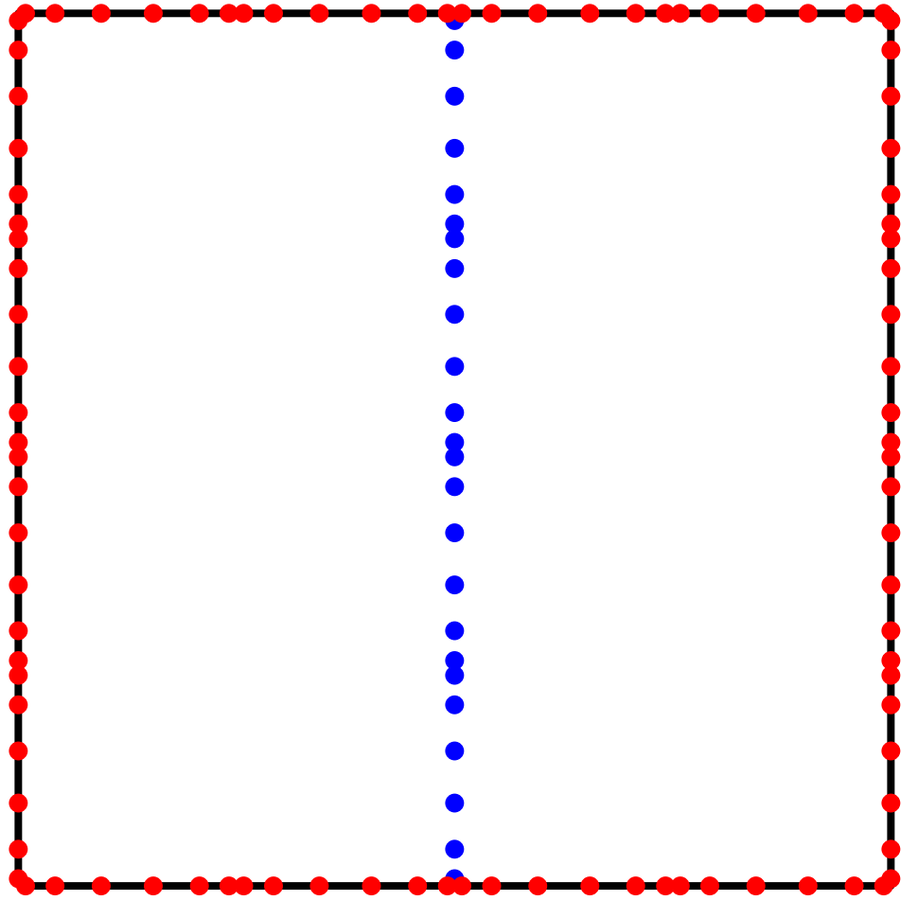
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



The Hierarchical Poincaré-Steklov Method

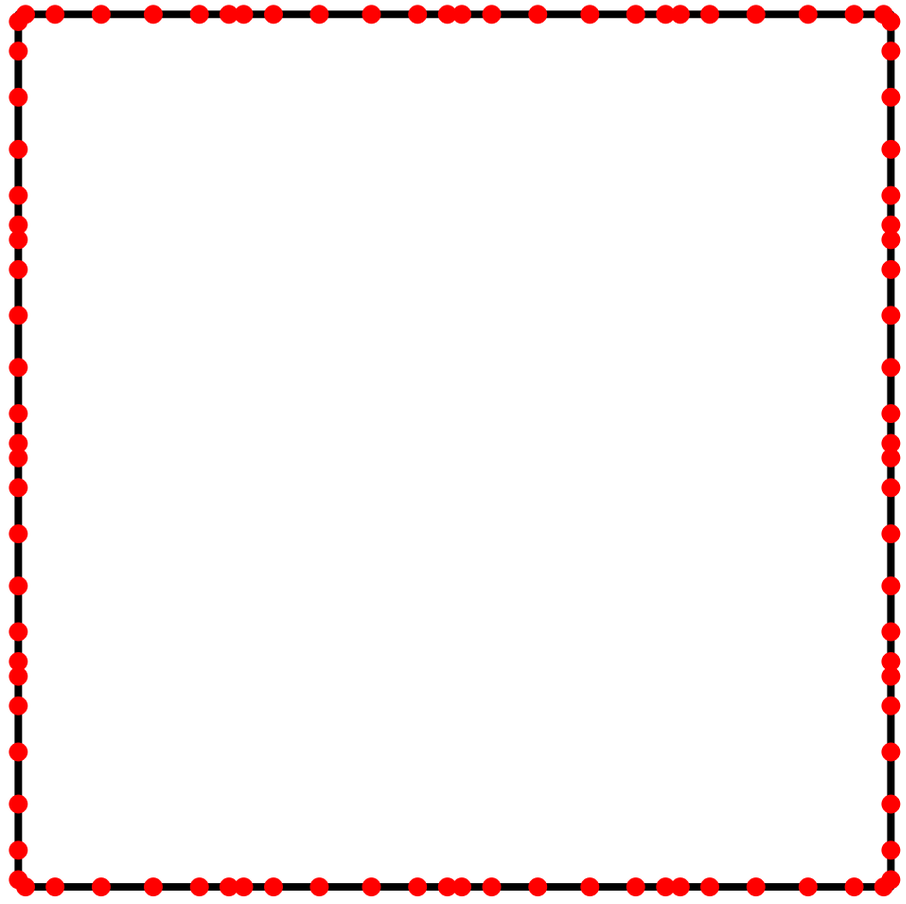
Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Upwards sweep: Merge boxes by pairs and eliminate the interior (blue) nodes.

To do this, use the computed DtN operators to enforce continuity of u and du/dn across interior boundaries. Compute the DtN operator for the larger box.



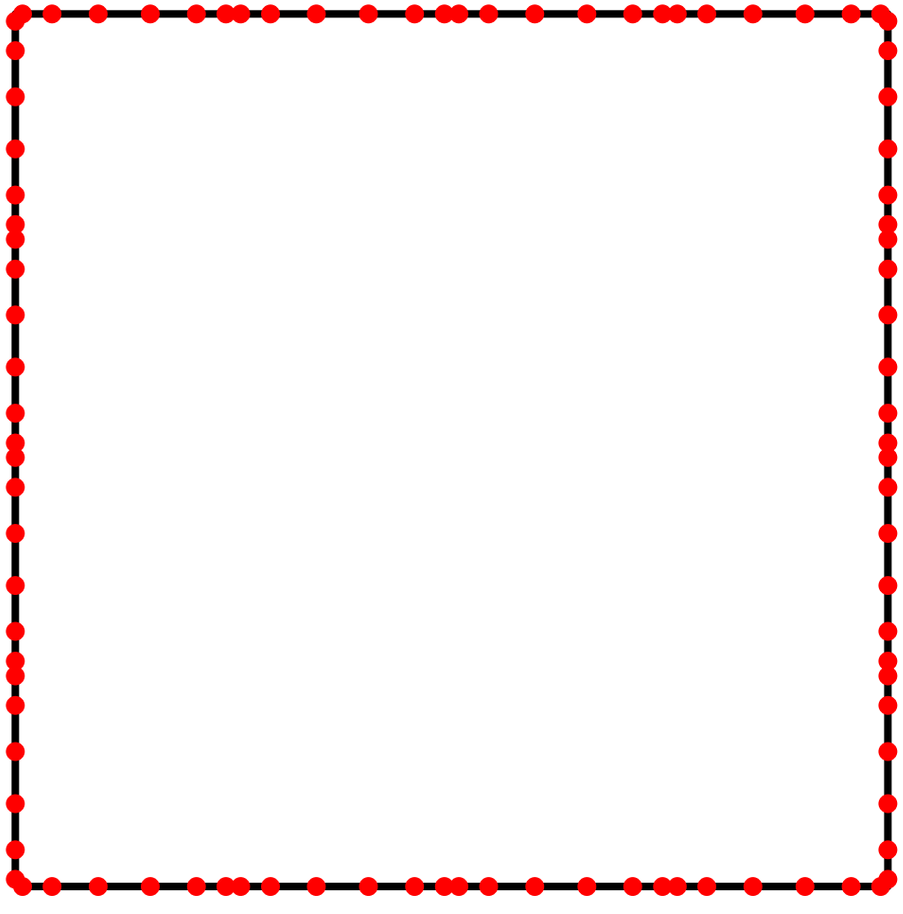
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Top level solve: Invert the DtN operator for the top level box.



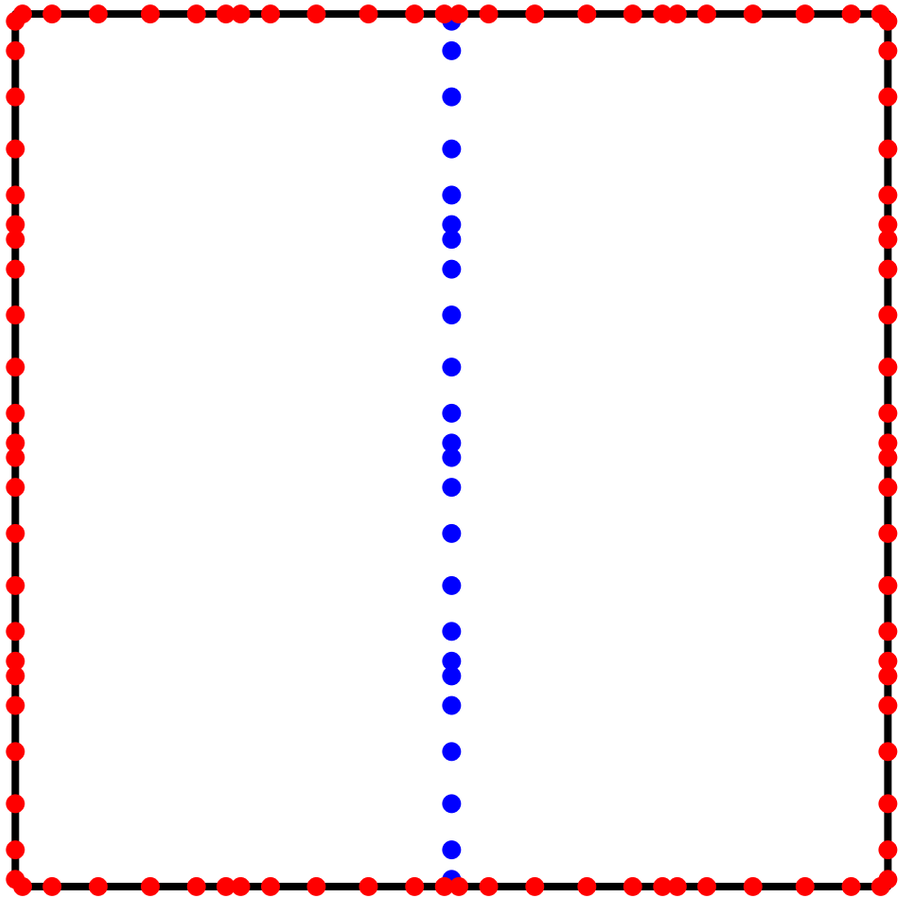
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.



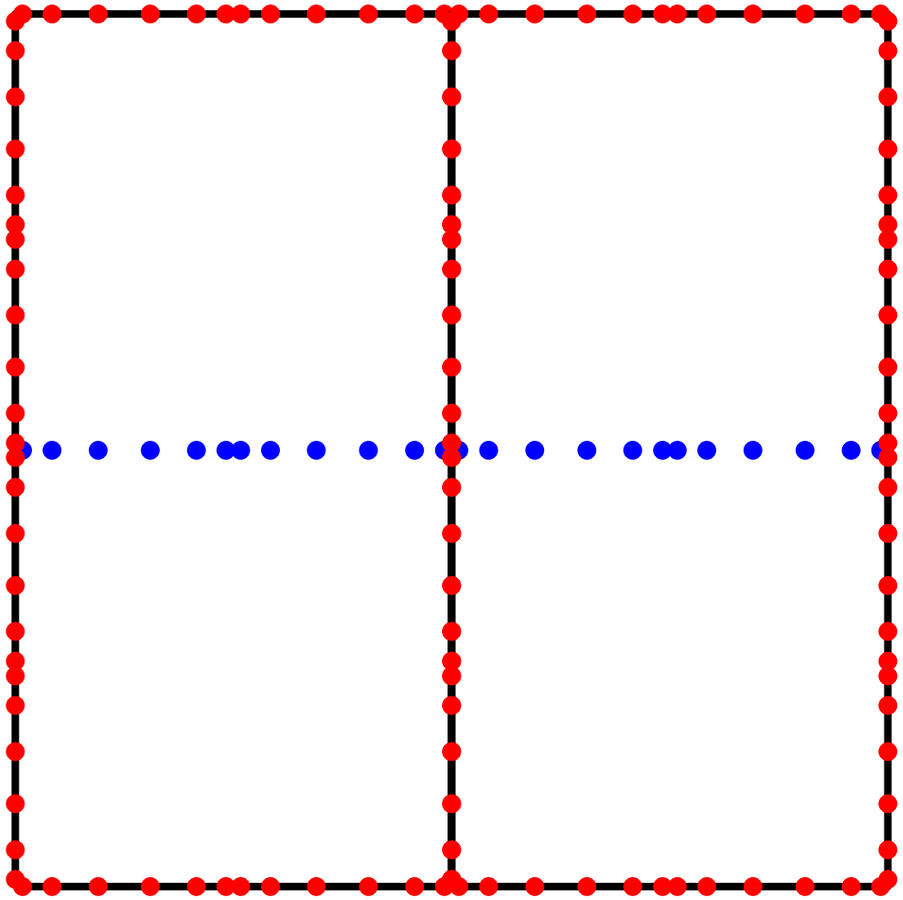
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.



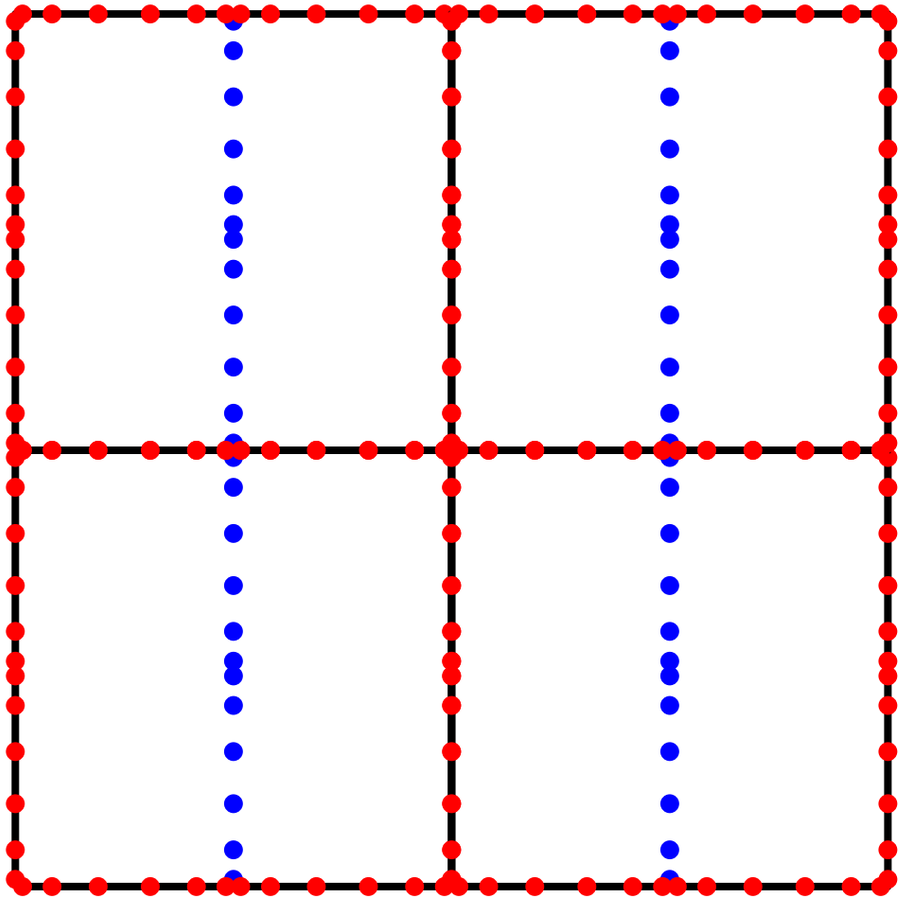
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.



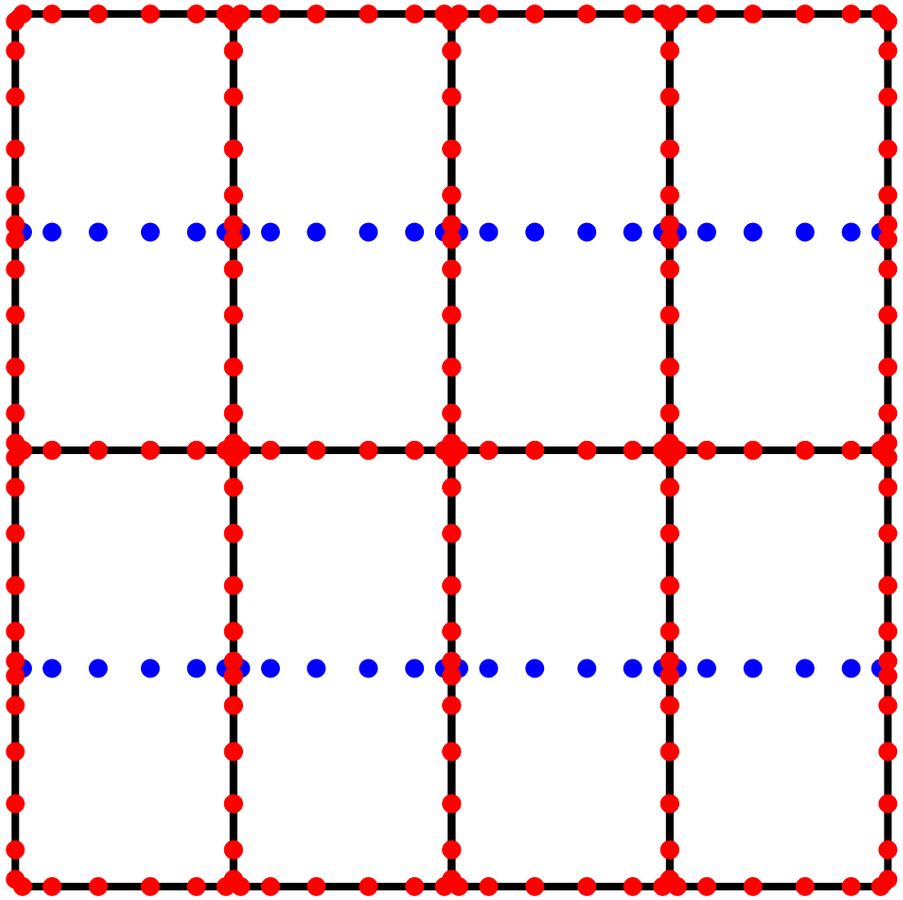
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.



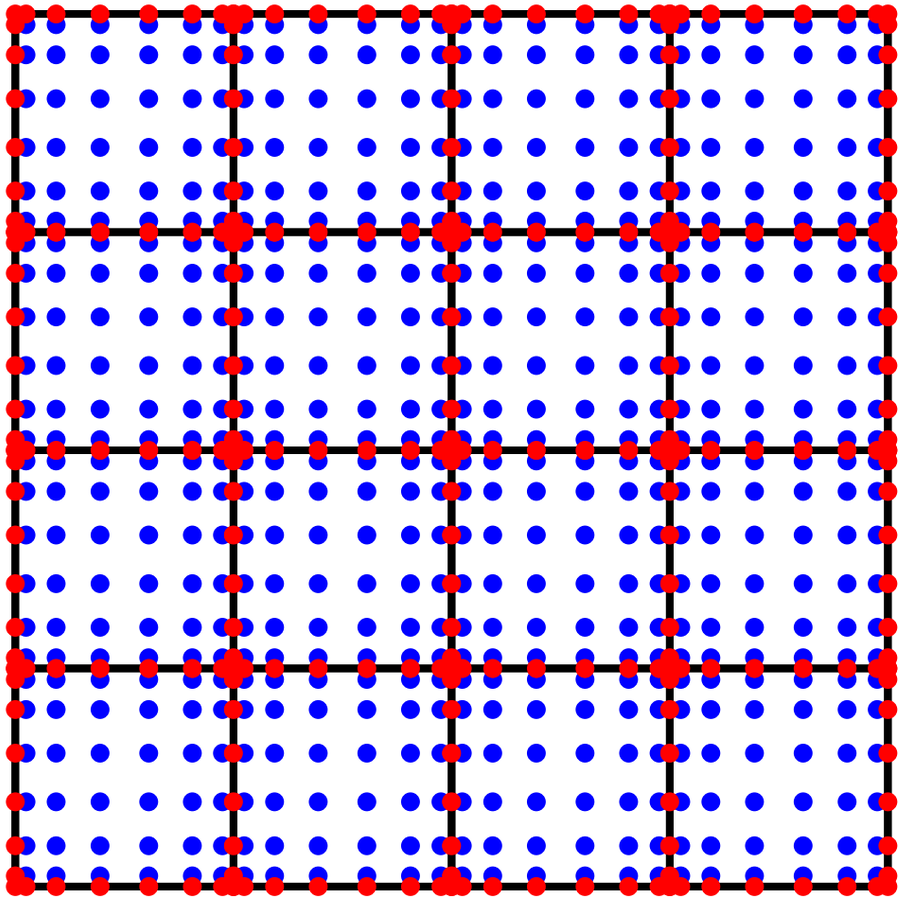
The Hierarchical Poincaré-Steklov Method

Model problem: Given f, g, and b, find u such that

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

where $\Omega = [0,1]^2$ is the unit square and $\Gamma = \partial\Omega$. We assume u is smooth.

Downwards sweep: We know u on the red nodes. We can use the computed DtN operators to reconstruct u on the blue nodes.



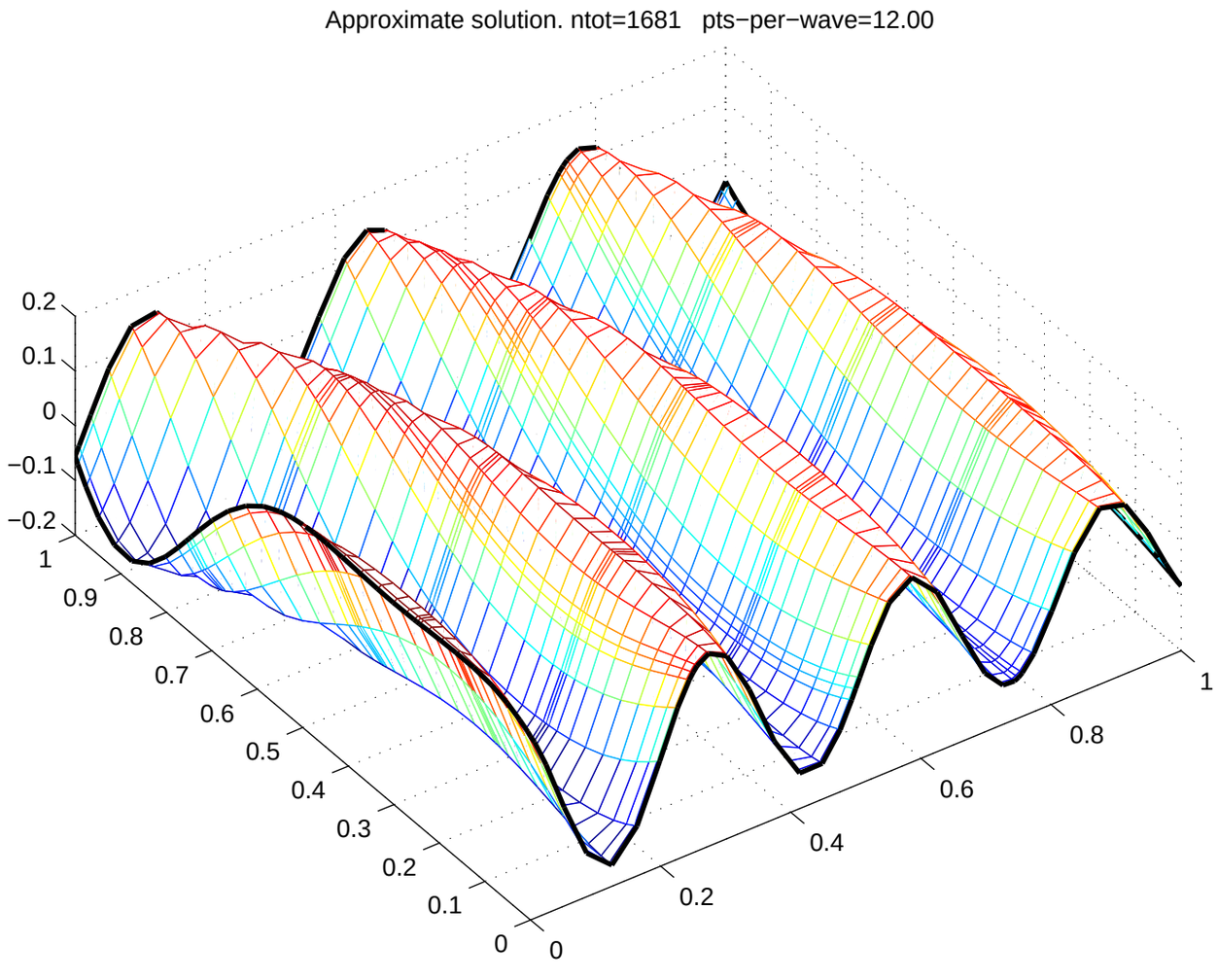
Hierarchical Poincaré-Steklov Method: numerical results

Set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.



Hierarchical Poincaré-Steklov Method: numerical results

Set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

The spectral computation on a leaf involves 21×21 points.

κ is chosen so that there are 12 points per wave-length.

p	N	N_{wave}	t_{build} (sec)	t_{solve} (sec)	E_{pot}	E_{grad}	M (MB)	M/N (reals/DOF)
21	6561	6.7	0.23	0.0011	2.56528e-10	1.01490e-08	4.4	87.1
21	25921	13.3	0.92	0.0044	5.24706e-10	4.44184e-08	18.8	95.2
21	103041	26.7	4.68	0.0173	9.49460e-10	1.56699e-07	80.8	102.7
21	410881	53.3	22.29	0.0727	1.21769e-09	3.99051e-07	344.9	110.0
21	1640961	106.7	99.20	0.2965	1.90502e-09	1.24859e-06	1467.2	117.2
21	6558721	213.3	551.32	20.9551	2.84554e-09	3.74616e-06	6218.7	124.3

Error is measured in sup-norm: $e = \max_{\boldsymbol{x} \in \Omega} |u(\boldsymbol{x}) - u_{\text{exact}}(\boldsymbol{x})|$.

Note 1: Translation invariance is *not* exploited.

Note 2: The times refer to a simple Matlab implementation executed on a $1k laptop.

Note 3: Keeping a fixed number of points per wave-length works well for this scheme!

Hierarchical Poincaré-Steklov Method: numerical results

Set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

The spectral computation on a leaf involves 41×41 points.

κ is chosen so that there are 12 points per wave-length.

p	N	N_{wave}	t_{build} (sec)	t_{solve} (sec)	E_{pot}	E_{grad}	M (MB)	M/N (reals/DOF)
41	6561	6.7	1.50	0.0025	9.88931e-14	3.46762e-12	7.9	157.5
41	25921	13.3	4.81	0.0041	1.58873e-13	1.12883e-11	32.9	166.4
41	103041	26.7	18.34	0.0162	3.95531e-13	5.51141e-11	137.1	174.4
41	410881	53.3	75.78	0.0672	3.89079e-13	1.03546e-10	570.2	181.9
41	1640961	106.7	332.12	0.2796	1.27317e-12	7.08201e-10	2368.3	189.2

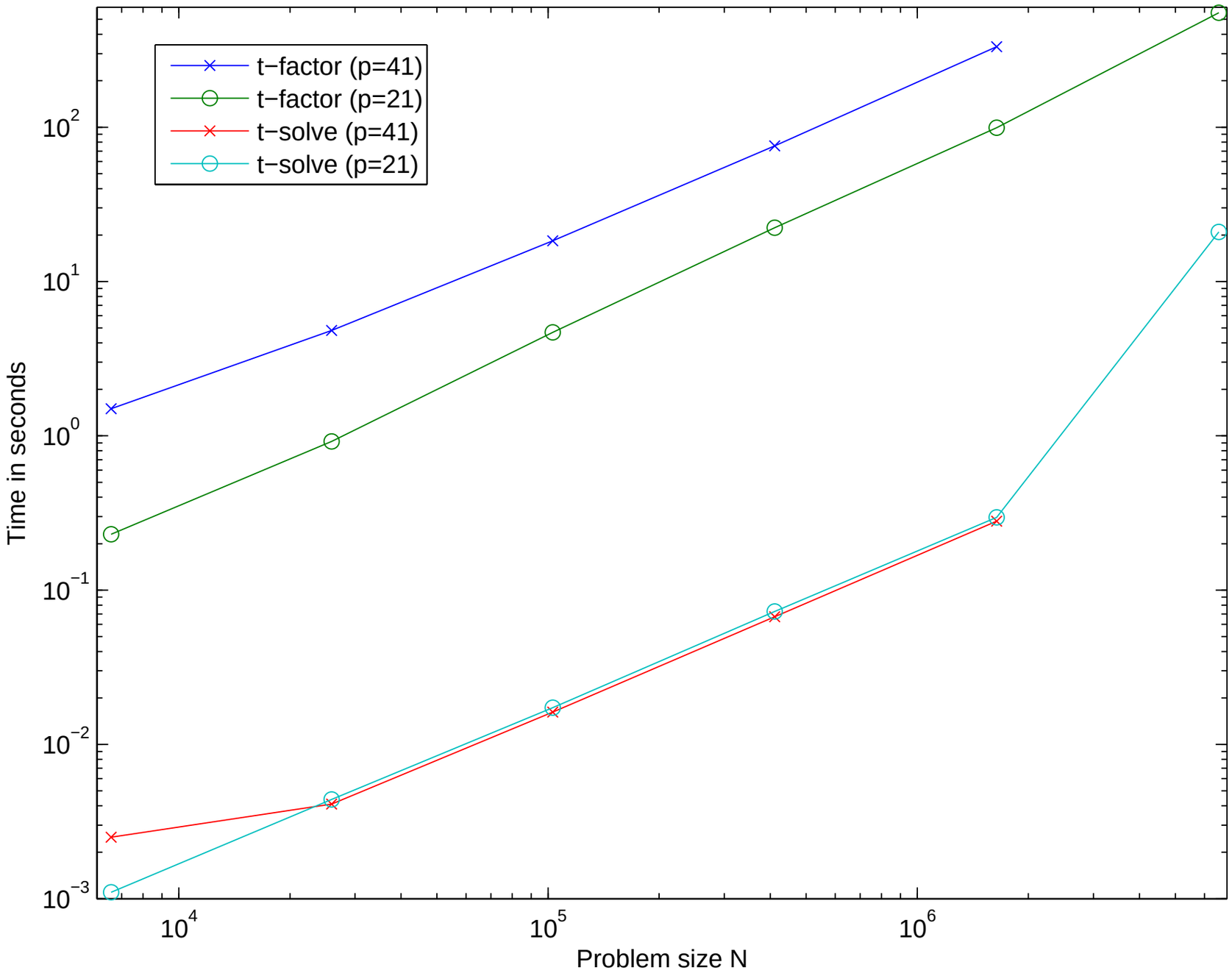
Error is measured in sup-norm: $e = \max_{\boldsymbol{x} \in \Omega} |u(\boldsymbol{x}) - u_{\text{exact}}(\boldsymbol{x})|$.

Note 1: Translation invariance is *not* exploited.

Note 2: The times refer to a simple Matlab implementation executed on a $1k laptop.

Note 3: Keeping a fixed number of points per wave-length works well for this scheme!

Spectral composite method: numerical results



The line t_{solve} scales perfectly linearly (until memory problems kick in), as expected.

Interesting: The line t_{build} also scales almost linearly. (Unexpectedly?) It turns out that t_{build} is dominated by the leaf computation; we have not yet hit the $O(N^{1.5})$ asymptotic.

Hierarchical Poincaré-Steklov Method: numerical results — variable coefficients

Now consider the variable coefficient problem

$$\begin{aligned} -\Delta u(\boldsymbol{x}) - \kappa^2 \big(1 - b(\boldsymbol{x})\big)\, u(\boldsymbol{x}) &= 0 && \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= f(\boldsymbol{x}) && \boldsymbol{x} \in \Gamma, \end{aligned}$$

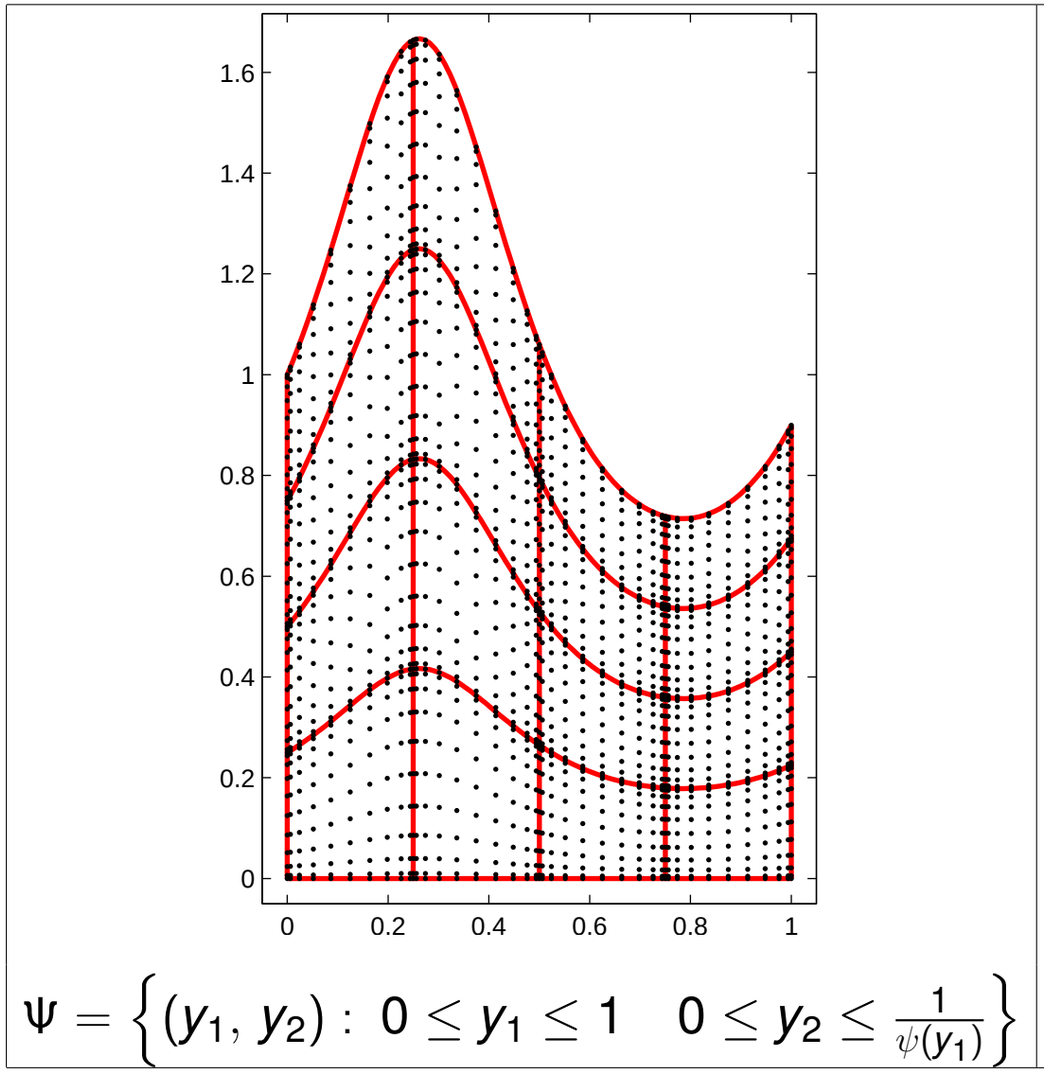
where $\Omega = [0,1]^2$, where $\Gamma = \partial\Omega$, and where $b(\boldsymbol{x}) = (\sin(4\pi x_1)\,\sin(4\pi x_2))^2$.

The Helmholtz parameter was kept fixed at $\kappa = 80$, corresponding to a domain size of 12.7×12.7 wave lengths. The boundary data was given by $f(\boldsymbol{x}) = \cos(8x_1)\,\big(1 - 2x_2\big)$.

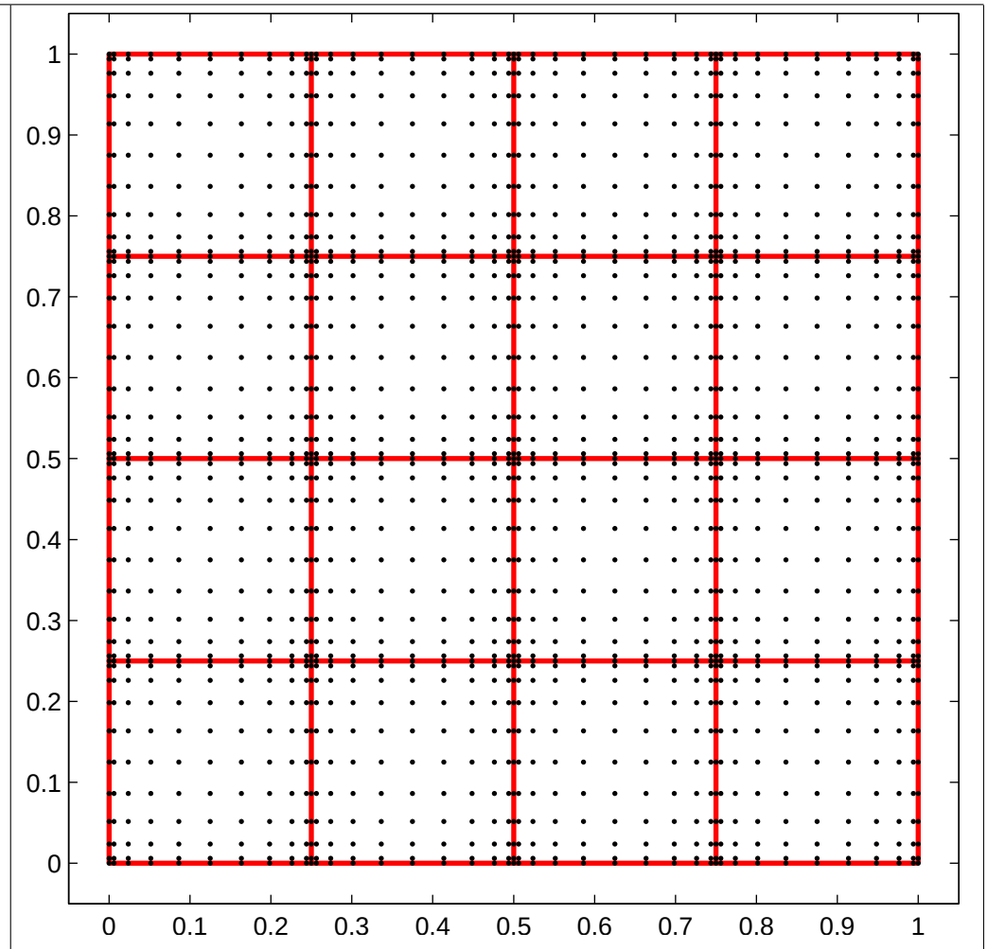
The error estimator $E_N^{\text{int}} = u_N(\hat{\boldsymbol{x}}) - u_{4N}(\hat{\boldsymbol{x}})$ where $\hat{\boldsymbol{x}} = (0.75,\, 0.25)$ is reported below:

p	N	pts per wave	$u_N(\hat{\boldsymbol{x}})$	E_N^{int}	$w_N(\hat{\boldsymbol{y}})$	E_N^{bnd}
21	6561	6.28	-2.448236804078803	-1.464e-03	-32991.4583727724	2.402e+02
21	25921	12.57	-2.446772430608166	7.976e-08	-33231.6118304666	5.984e-03
21	103041	25.13	-2.446772510369452	5.893e-11	-33231.6178142514	-5.463e-06
21	410881	50.27	-2.446772510428384	2.957e-10	-33231.6178087887	-2.792e-05
21	1640961	100.53	-2.446772510724068		-33231.6177808723	
41	6561	6.28	-2.446803898373796	-3.139e-05	-33233.0037457220	-1.386e+00
41	25921	12.57	-2.446772510320572	1.234e-10	-33231.6179029824	-8.940e-05
41	103041	25.13	-2.446772510443995	2.888e-11	-33231.6178135860	-1.273e-05
41	410881	50.27	-2.446772510472872	7.731e-11	-33231.6178008533	-4.668e-05
41	1640961	100.53	-2.446772510550181		-33231.6177541722	

A curved domain



$$\Psi = \left\{ (y_1, y_2) : \ 0 \le y_1 \le 1 \quad 0 \le y_2 \le \frac{1}{\psi(y_1)} \right\}$$



$$\Omega = [0, 1]^2$$

Consider a *curved domain* Ψ as shown above and the equation

$$\begin{cases} -\Delta u(\boldsymbol{y}) - \kappa^2 u(\boldsymbol{y}) = 0 & \boldsymbol{y} \in \Psi, \\ u(\boldsymbol{y}) = f(\boldsymbol{y}) & \boldsymbol{y} \in \partial\Psi. \end{cases} \tag{4}$$

The reparameterization is $y_1 = x_1$ and $y_2 = \psi(y_1)\, y_2$, and so the Helmholtz equation (4) takes the form

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{2\psi'(x_1)\, x_2}{\psi(x_1)} \frac{\partial^2 u}{\partial x_1 \partial x_2} + \left(\frac{x_2^2 \psi'(x_1)^2}{\psi(x_1)^2} + \psi(x_1)^2 \right) \frac{\partial^2 u}{\partial x_2^2} + \frac{x_2 \psi''(x_1)}{\psi(x_1)} \frac{\partial u}{\partial x_2} + k^2 u = 0, \qquad \boldsymbol{x} \in \Omega.$$

Numerical results for curved domain

The equation is (constant coefficient) Helmholtz on a domain of size 35×50 wave lengths.

	Exact solution known		*Dirichlet data* $f = 1$		
N	E_{pot}	E_{grad}	$E_N^{(1)}$	$E_N^{(2)}$	$E_N^{(3)}$
25921	2.12685e+02	3.55772e+04	2.24618e-01	4.99854e-01	6.69023e-01
103041	3.29130e-01	5.89976e+01	1.10143e-02	5.28238e-03	6.14890e-02
410881	1.40813e-05	1.98907e-03	4.57900e-06	2.18438e-06	1.13415e-05
1640961	7.22959e-10	1.17852e-07	5.12914e-07	1.67971e-06	4.97764e-06
3690241	1.63144e-09	2.26204e-07	—	—	—

Hierarchical Poincaré-Steklov Method: Free space scattering

Consider the acoustic scattering problem

$$\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u_{\text{out}}(\boldsymbol{x}) = -\kappa^2 b(\boldsymbol{x})\, u_{\text{in}}(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^2 \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\left(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - i\kappa\, u_{\text{out}}(\boldsymbol{x})\right) = 0 \end{cases}$$

Suppose that b is a smooth scattering potential such that for some rectangle Ω, we have

$$\text{support}(b) \subset \Omega.$$

We also suppose that u_{in} satisfies

$$-\Delta u_{\text{in}}(\boldsymbol{x}) - \kappa^2 u_{\text{in}}(\boldsymbol{x}) = 0, \qquad \boldsymbol{x} \in \Omega.$$

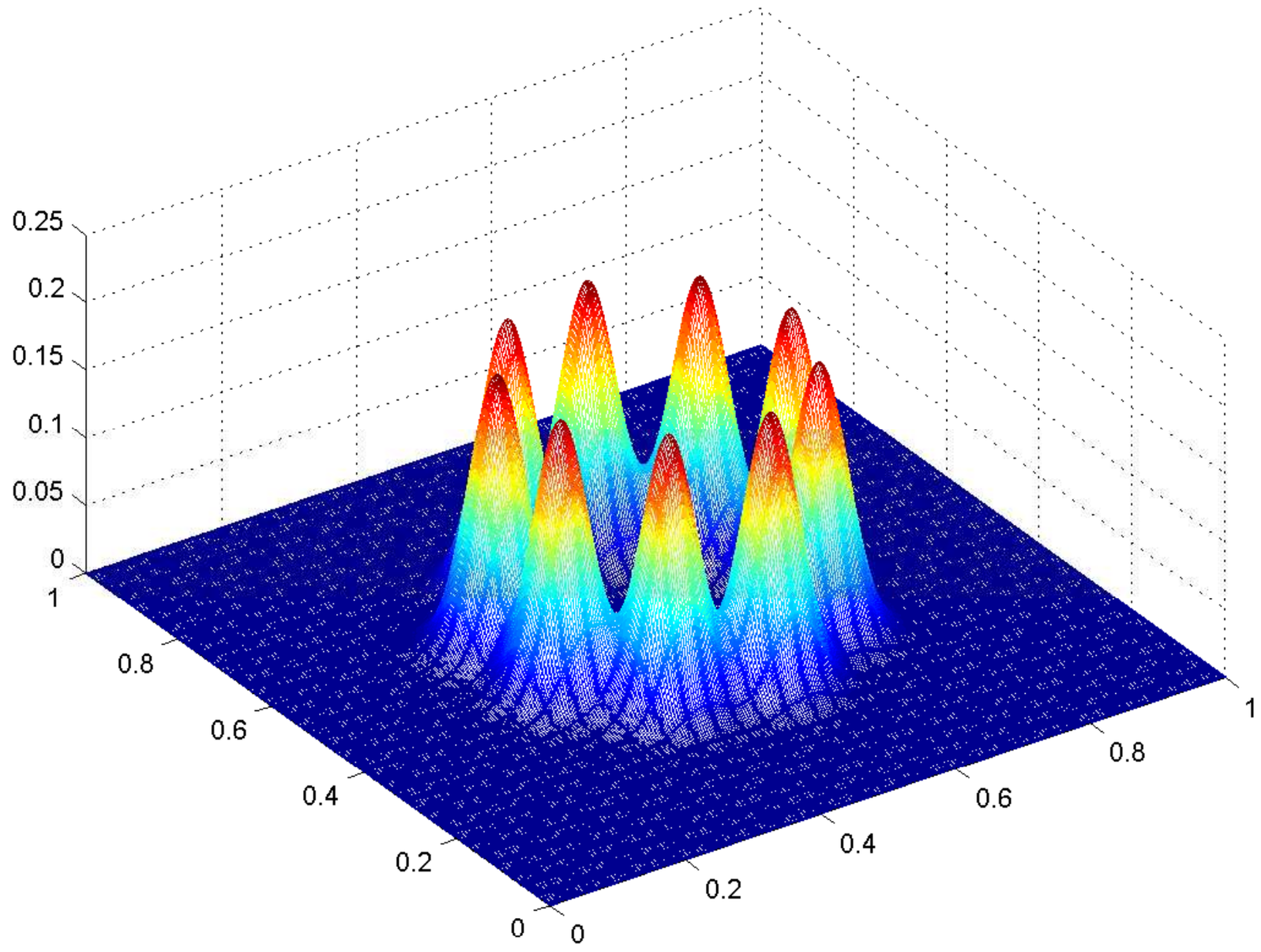
Solution strategy ("FEM-BEM coupling"):

1. Use HPS method to construct the DtN map for the variable coefficient problem in Ω.
2. Use boundary integral equation techniques to find the DtN map for the constant coefficient problem on Ω^{c}.
3. Glue the two DtN maps together and you're all set!

Joint work with Adrianna Gillman and Alex Barnett.

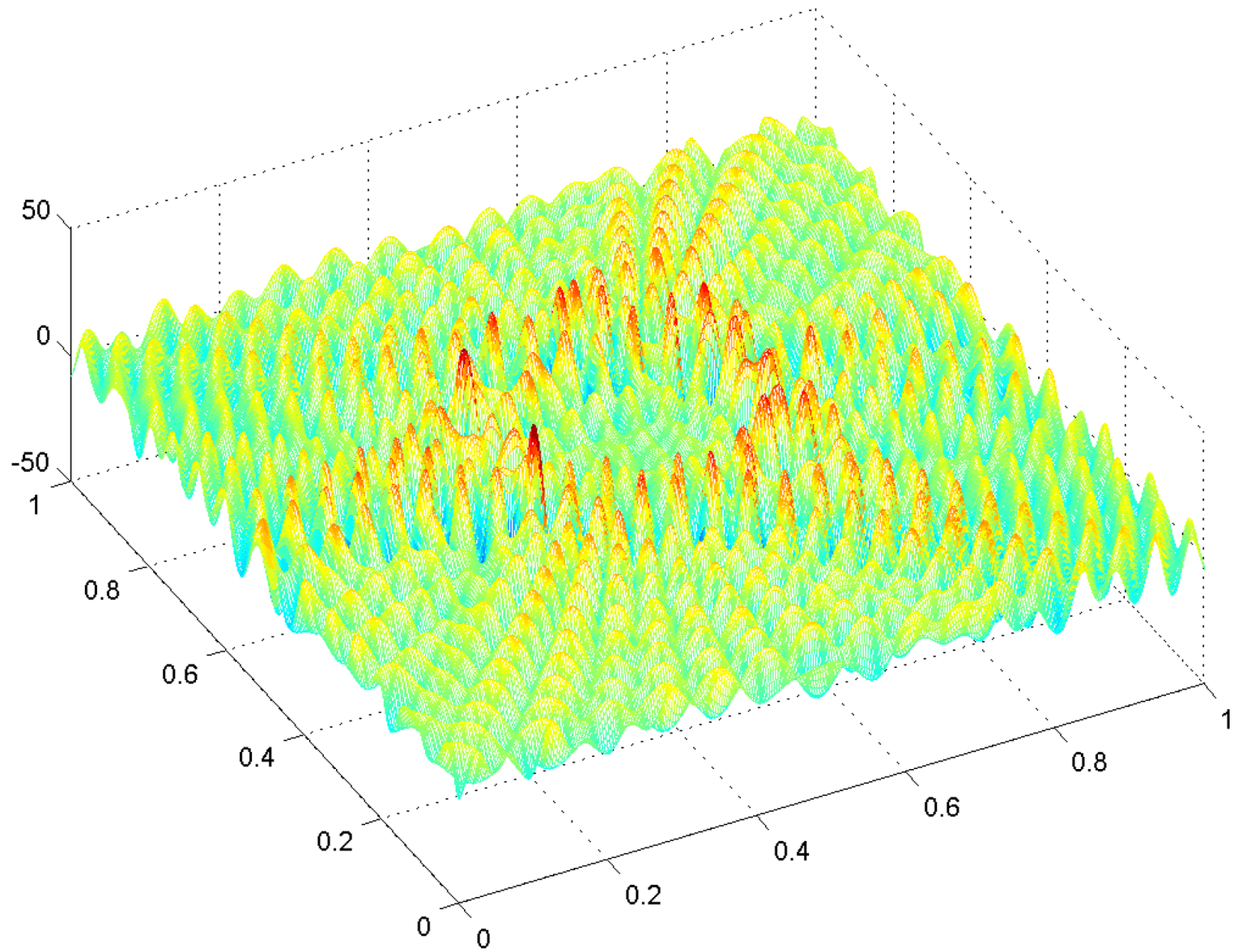
Example: Free space scattering $\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - \kappa^2 (1 - b(\boldsymbol{x}))\, u_{\text{out}}(\boldsymbol{x}) = -\kappa^2\, b(\boldsymbol{x})\, u_{\text{in}}(\boldsymbol{x}) \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - i\kappa\, u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$

The scattering potential b



Example: Free space scattering $\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - \kappa^2 (1 - b(\boldsymbol{x}))\, u_{\text{out}}(\boldsymbol{x}) = -\kappa^2\, b(\boldsymbol{x})\, u_{\text{in}}(\boldsymbol{x}) \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - i\kappa\, u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$

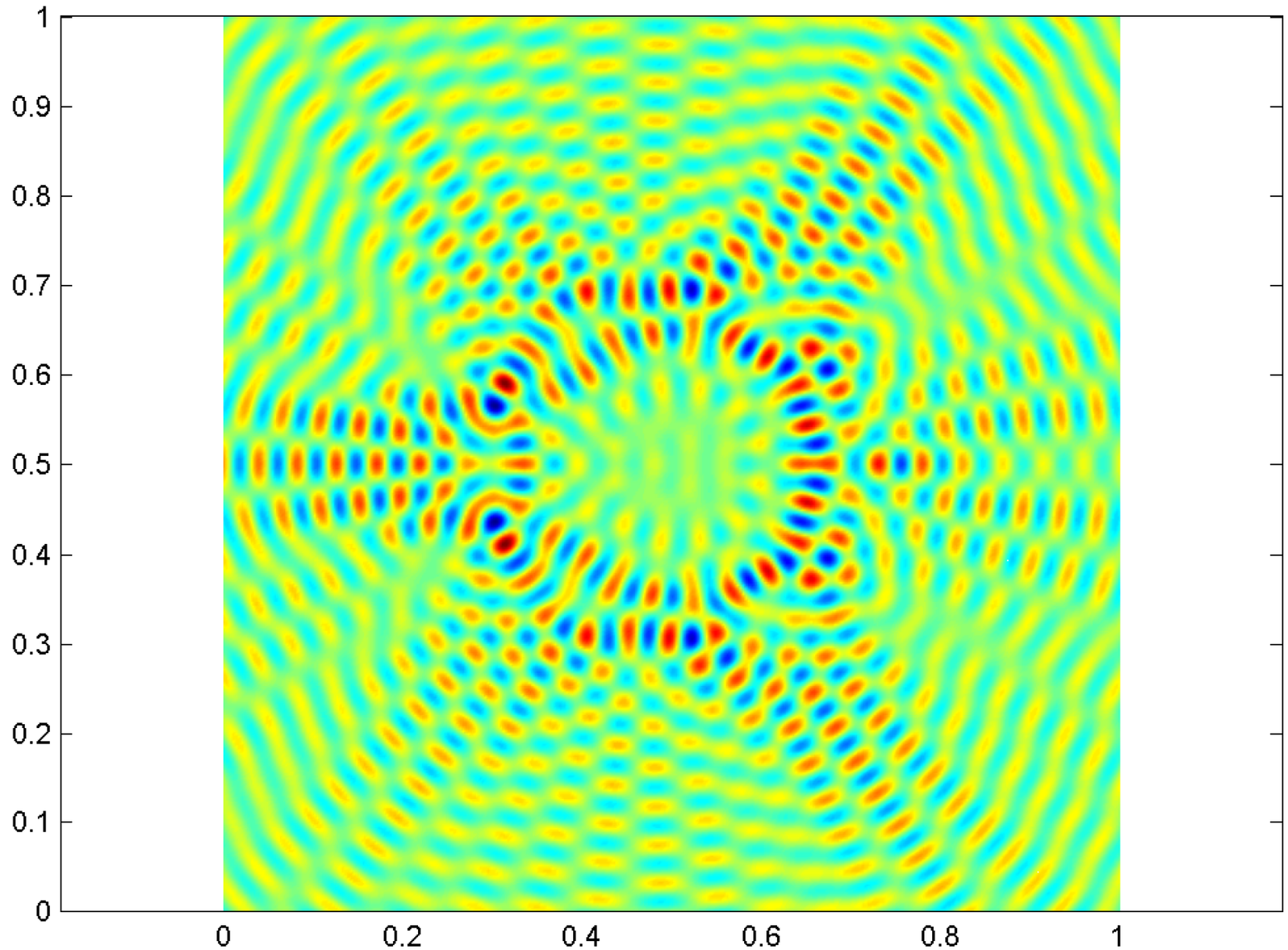
The outgoing field u_{out} (resulting from an incoming plane wave $u_{\text{in}}(x) = \cos(\kappa\, x_1)$)



$N = 231\,361$ $T_{\text{build}} = 7.2$ sec $T_{\text{solve}} = 0.06$ sec $E \approx 10^{-7}$ (estimated)

Example: Free space scattering $\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - \kappa^2 (1 - b(\boldsymbol{x}))\, u_{\text{out}}(\boldsymbol{x}) = -\kappa^2\, b(\boldsymbol{x})\, u_{\text{in}}(\boldsymbol{x}) \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\left(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - i\kappa\, u_{\text{out}}(\boldsymbol{x})\right) = 0 \end{cases}$

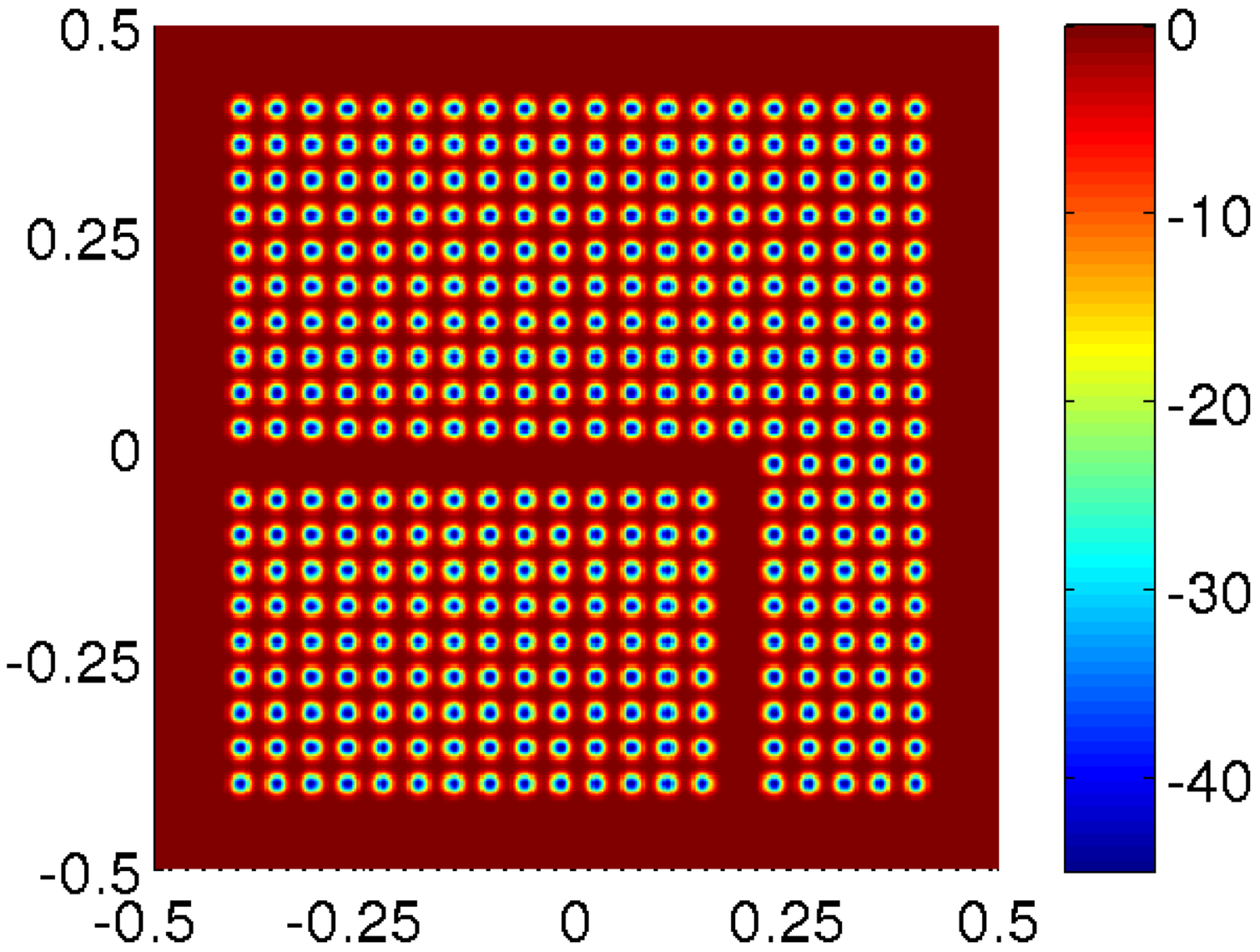
The outgoing field u_{out} (resulting from an incoming plane wave $u_{\text{in}}(x) = \cos(\kappa\, x_1)$)



$N = 231\,361$ $T_{\text{build}} = 7.2$ sec $T_{\text{solve}} = 0.06$ sec $E \approx 10^{-7}$ (estimated)

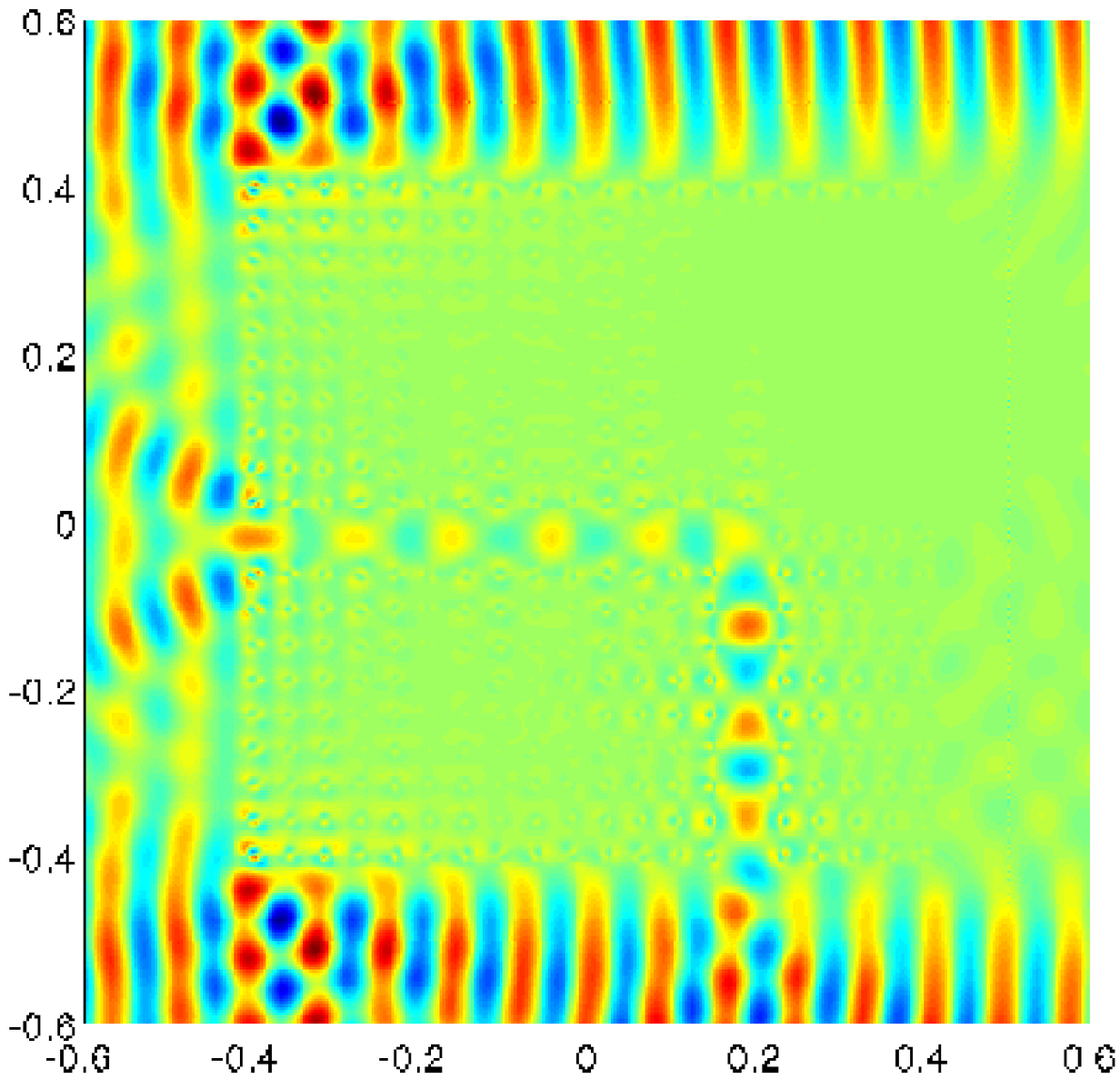
Example: Free space scattering $\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - \kappa^2 (1 - b(\boldsymbol{x}))\, u_{\text{out}}(\boldsymbol{x}) = -\kappa^2\, b(\boldsymbol{x})\, u_{\text{in}}(\boldsymbol{x}) \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\left(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - i\kappa\, u_{\text{out}}(\boldsymbol{x})\right) = 0 \end{cases}$

The scattering potential b — now a photonic crystal with a wave guide.



Example: Free space scattering $\begin{cases} -\Delta u_{\text{out}}(\boldsymbol{x}) - \kappa^2 (1 - b(\boldsymbol{x}))\, u_{\text{out}}(\boldsymbol{x}) = -\kappa^2\, b(\boldsymbol{x})\, u_{\text{in}}(\boldsymbol{x}) \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\text{out}}(\boldsymbol{x}) - i\kappa\, u_{\text{out}}(\boldsymbol{x})\big) = 0 \end{cases}$

The total field $u = u_{\text{in}} + u_{\text{out}}$ (resulting from an incoming plane wave $u_{\text{in}}(x) = \cos(\kappa x_1)$).



Recall: The method as presented relies on a hierarchical construction of Dirichlet-to-Neumann operators for every box in a hierarchical tree.

Problem! The interior Helmholtz equation may encounter *resonances* — even for zero Dirichlet data, there may be non-trivial solutions.

Conceptual problem : The DtN operator does not always exist.

Practical problem: The numerical DtN operator can be very ill-conditioned.

Solution: Rather than the *Dirichlet-to-Neumann map*

$$T: \quad u|_\Gamma \quad \mapsto \quad \left.\frac{\partial u}{\partial n}\right|_\Gamma$$

consider the *impedance map*

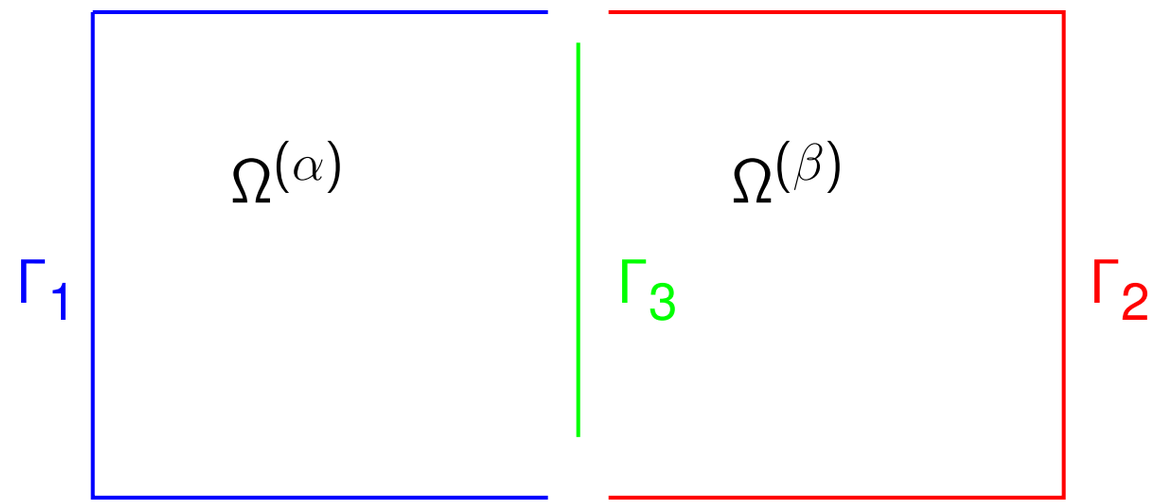
$$E: \quad u|_\Gamma + \mathrm{i}\left.\frac{\partial u}{\partial n}\right|_\Gamma \quad \mapsto \quad u|_\Gamma - \mathrm{i}\left.\frac{\partial u}{\partial n}\right|_\Gamma$$

The impedance map exists for every wave-number, and is a unitary map.

Joint work with Alexander Barnett (Dartmouth) and Adrianna Gillman (Rice).

The build stage can be accelerated to optimal $O(N)$ complexity:

Consider the merge of two patches $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ with boundaries Γ_1, Γ_2, Γ_3:



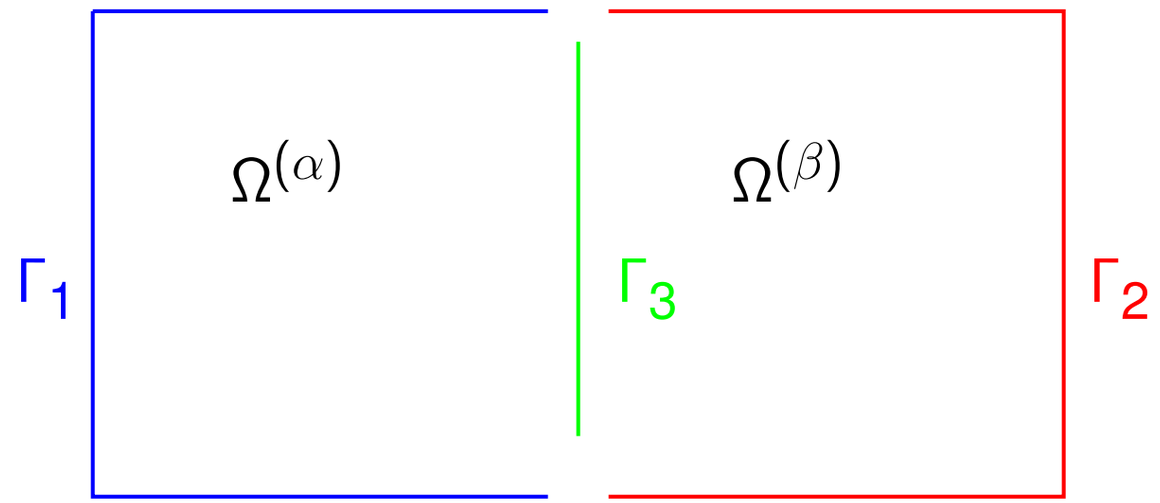
In the composite spectral method we have

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}^{(\alpha)}_{1,1} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^{(\beta)}_{2,2} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{T}^{(\alpha)}_{1,3} \\ \mathbf{T}^{(\beta)}_{2,3} \end{bmatrix} \left(\mathbf{T}^{(\alpha)}_{3,3} - \mathbf{T}^{(\beta)}_{3,3}\right)^{-1} \left[-\mathbf{T}^{(\alpha)}_{3,1} \mid \mathbf{T}^{(\beta)}_{3,2}\right]}_{\text{low rank update!}}.$$

There is more structure!

The build stage can be accelerated to optimal $O(N)$ complexity:

Consider the merge of two patches $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ with boundaries Γ_1, Γ_2, Γ_3:



In the composite spectral method we have

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}^{(\alpha)}_{1,1} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}^{(\beta)}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathbf{T}^{(\alpha)}_{1,3} \\ \mathbf{T}^{(\beta)}_{2,3} \end{bmatrix} \left(\mathbf{T}^{(\alpha)}_{3,3} - \mathbf{T}^{(\beta)}_{3,3}\right)^{-1} \left[-\mathbf{T}^{(\alpha)}_{3,1} \mid \mathbf{T}^{(\beta)}_{3,2}\right].$$

There is more structure:

- The blue terms are of low numerical rank (say rank 40 to precision 10^{-10}).
- The red terms are "hierarchically block separable" matrices.
  (Their off-diagonal blocks have low rank, cf. $\mathcal{H}$-matrices, etc).

The bottom line is that *the solution operators can be built in optimal $O(N)$ time.*

(Not true when N is scaled to the wave-length for Helmholtz-type problems.)

Joint work with Adrianna Gillman.

Claim: Matrices with low-rank off-diagonal blocks can be inverted/multiplied/...rapidly.

As an example, consider a 2×2 blocked matrix of size $2n \times 2n$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}.$$

Suppose the off-diagonal blocks are rank-deficient

$$\underset{n \times n}{\mathbf{A}_{12}} = \underset{n \times k}{\mathbf{U}_1} \; \underset{k \times k}{\tilde{\mathbf{A}}_{12}} \; \underset{k \times n}{\mathbf{V}_2^*} \quad \text{and} \quad \underset{n \times n}{\mathbf{A}_{21}} = \underset{n \times k}{\mathbf{U}_2} \; \underset{k \times k}{\tilde{\mathbf{A}}_{21}} \; \underset{k \times n}{\mathbf{V}_1^*},$$

where $k \ll n$. We can then write $\mathbf{A}$ as follows

$$\mathbf{A} = \underbrace{\begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix}}_{\text{“easy” to invert}} + \underbrace{\begin{bmatrix} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{12} \\ \tilde{\mathbf{A}}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \end{bmatrix}}_{\text{low rank}}.$$

Recall the Woodbury formula

$$\left(\mathbf{D} + \mathbf{U}\tilde{\mathbf{A}}\mathbf{V}^*\right)^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{U}\left(\tilde{\mathbf{A}} + \mathbf{V}^*\mathbf{D}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}^*\mathbf{D}^{-1}.$$

Applying the Woodbury formula, we find, with $\mathbf{S}_{11} = \mathbf{V}_1^*\mathbf{A}_{11}^{-1}\mathbf{U}_1$ and $\mathbf{S}_2 = \mathbf{V}_2^*\mathbf{A}_{22}^{-1}\mathbf{U}_2$,

$$\underset{2n \times 2n}{\mathbf{A}^{-1}} = \underset{2n \times 2n}{\begin{bmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \end{bmatrix}} + \underset{2n \times 2k}{\begin{bmatrix} \mathbf{A}_{11}^{-1}\mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1}\mathbf{U}_2 \end{bmatrix}} \underset{2k \times 2k}{\begin{bmatrix} \mathbf{S}_1 & \tilde{\mathbf{A}}_{12} \\ \tilde{\mathbf{A}}_{21} & \mathbf{S}_2 \end{bmatrix}^{-1}} \underset{2k \times 2n}{\begin{bmatrix} \mathbf{V}_1^*\mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^*\mathbf{A}_{22}^{-1} \end{bmatrix}}.$$

Now suppose $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ have the same structure, and recurse.

Hierarchical Poincaré-Steklov Method: numerical results — $O(N)$ version

Problem	N	T_{build}	T_{solve}	MB
Laplace	1.7e6	91.68	0.34	1611.19
	6.9e6	371.15	1.803	6557.27
	2.8e7	1661.97	6.97	26503.29
	1.1e8	6894.31	30.67	106731.61
Helmholtz I	1.7e6	62.07	0.202	1611.41
	6.9e6	363.19	1.755	6557.12
	2.8e7	1677.92	6.92	26503.41
	1.1e8	7584.65	31.85	106738.85
Helmholtz II	1.7e6	93.96	0.29	1827.72
	6.9e6	525.92	2.13	7151.60
	2.8e7	2033.91	8.59	27985.41
Helmholtz III	1.7e6	105.58	0.44	1712.11
	6.9e6	510.37	2.085	7157.47
	2.8e7	2714.86	10.63	29632.89

(About six accurate digits in solution.)

Thanks to A. Barnett for use of a work-station!

Hierarchical Poincaré-Steklov Method: numerical results — $O(N)$ version

Problem	$\epsilon = 10^{-7}$		$\epsilon = 10^{-10}$		$\epsilon = 10^{-12}$	
	E_{pot}	E_{grad}	E_{pot}	E_{grad}	E_{pot}	E_{grad}
Laplace	6.54e-05	1.07e-03	2.91e-08	5.52e-07	1.36e-10	8.07e-09
Helmholtz I	7.45e-06	6.56e-04	5.06e-09	4.89e-07	1.38e-10	8.21e-09
Helmholtz II	6.68e-07	3.27e-04	1.42e-09	8.01e-07	8.59e-11	4.12e-08
Helmholtz III	7.40e-07	4.16e-04	2.92e-07	5.36e-06	1.66e-09	8.02e-08

Hierarchical Poincaré-Steklov Method: numerical results — $O(N)$ version

$$(5) \qquad \begin{cases} -\Delta u(\boldsymbol{x}) - c_1(\boldsymbol{x})\,\partial_1 u(\boldsymbol{x}) - c_2(\boldsymbol{x})\,\partial_2 u(\boldsymbol{x}) - c(\boldsymbol{x})\,u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

Laplace Let $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = c(\boldsymbol{x}) = 0$ in (5).

Helmholtz I Let $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = 0$, and $c(\boldsymbol{x}) = \kappa^2$ where $\kappa = 80$ in (5). This represents a vibration problem on a domain Ω of size roughly 12×12 wave-lengths. (Recall that the wave-length is given by $\lambda = \frac{2\pi}{\kappa}$.)

Helmholtz II Let $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = 0$, and $c(\boldsymbol{x}) = \kappa^2$ where $\kappa = 640$ in (5). This corresponds to a domain of size roughly 102×102 wave-lengths.

Helmholtz III We again set $c_1(\boldsymbol{x}) = c_2(\boldsymbol{x}) = 0$, and $c(\boldsymbol{x}) = \kappa^2$ in (5), but now we let κ grow as the number of discretization points grows to maintain a constant 12 points per wavelength.

Hierarchical Poincaré-Steklov Method: numerical results — $O(N)$ version in 3D

Before showing the results from 3D ... some programming notes ...

- *These results are very tentative ... code recently completed ...*
- *Timings for the BUILD stage are very bad ... can be greatly improved ... I think ...*
- *Memory requirements are bad (by current standards). Can be improved some.*
- *Solve time is excellent! And can be improved!*

Hierarchical Poincaré-Steklov Method: numerical results — $O(N)$ version in 3D

Set $\Omega = [0,1]^3$ and $\Gamma = \partial\Omega$. Consider the problem

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

We pick f as the restriction of a field from a point source, $\boldsymbol{x} \mapsto |\boldsymbol{x} - \hat{\boldsymbol{x}}|^{-1}$.

We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = |\boldsymbol{x} - \hat{\boldsymbol{x}}|^{-1}$.

N_{tot}	R (GB)	T_{build} (sec)	T_{solve} (sec)	E^{∞}	E^{rel}
4 913	0.04	0.97	0.004	1.20e-06	3.38e-05
35 937	0.52	20.34	0.032	1.45e-08	4.08e-07
274 625	6.33	522.78	0.24	5.48e-08	1.54e-07
2 146 689	76.59	17103.21 ($\approx$ 5h)	1121.0	6.51e-09	1.83e-07

Hierarchical Poincaré-Steklov Method: numerical results — $O(N)$ version in 3D

Set $\Omega = [0,1]^3$ and $\Gamma = \partial\Omega$. Consider the problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

We pick f as the restriction of a wave from a point source, $\boldsymbol{x} \mapsto Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

We then know the exact solution, $u_{\text{exact}}(\boldsymbol{x}) = Y_0(\kappa|\boldsymbol{x} - \hat{\boldsymbol{x}}|)$.

N_{tot}	N_{Gauss}	Memory (GB)	T_{build} (sec)	T_{solve} (sec)	E^{∞}	E^{rel}
274 625	9	8.65	1034.3	0.2	1.34e+00	3.76e+01
531 441	11	18.40	2910.6	0.5	1.70e-01	4.78e+00
912 673	13	34.55	7573.7	1.1	7.50e-03	2.11e-01
1 442 897	15	59.53	14161.1	2.8	9.45e-04	2.65e-02
2 146 689	17	97.73	25859.3	978.7	5.26e-05	1.48e-03

Results for solving Helmholtz equation with compression parameter $\epsilon = 10^{-5}$ *with* $20 \times 20 \times 20$ *wavelength across the domain.*

Note: In all cases, *application of the solution operator is extremely fast.*

Observation 1: The direct solver can be used to accelerate *implicit time-stepping schemes* for parabolic PDEs. As a toy example, consider:

$$\begin{cases} -\dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = -\Delta u, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x},t) = f(\boldsymbol{x},t) & \boldsymbol{x} \in \Gamma, \\ u(\boldsymbol{x},0) = h(\boldsymbol{x}) & \boldsymbol{x} \in \Omega. \end{cases}$$

Say, for simplicity, that we use backwards Euler to discretize in time, with

$$\frac{\partial u^n}{\partial t} \approx \frac{1}{k}\left(u^n - e^{n-1}\right).$$

Then for each time-step we need to solve

$$\begin{cases} -\Delta u^n + \dfrac{1}{k}u^n = \dfrac{1}{k}u^{n-1}, & \Omega, \\ u^n = f^n & \Gamma. \end{cases}$$

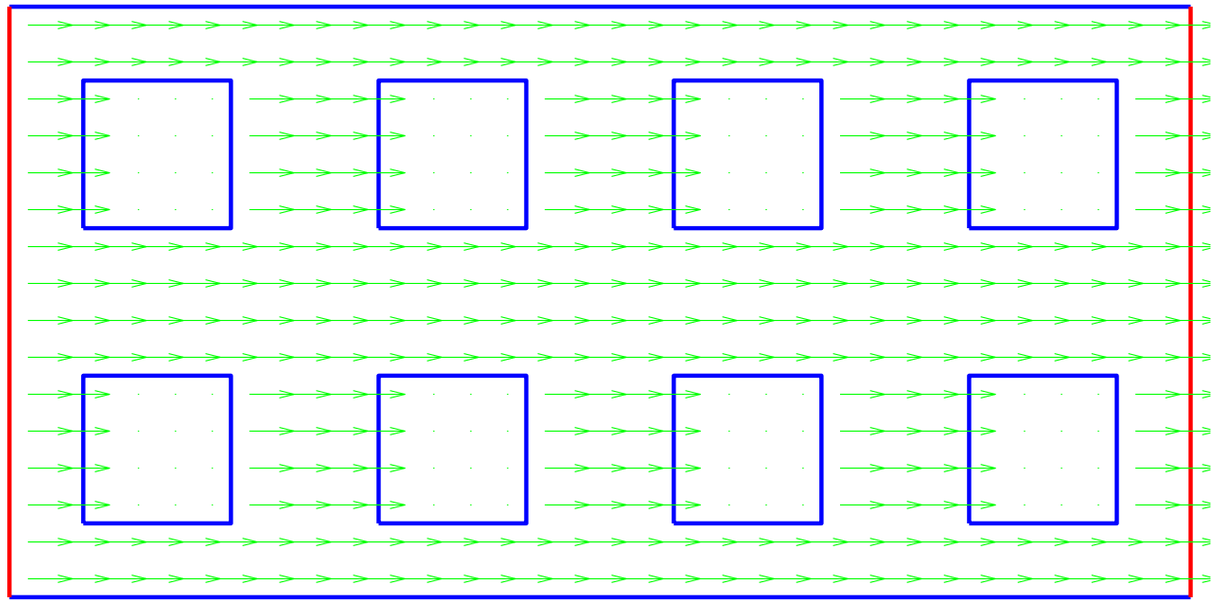
This is very well suited for our direct solver.

Current work: Investigate stability with better time-stepping schemes (specifically ESDIRK). Numerical experiments are very promising. Extension to *Stokes*, low Reynolds number *Navier-Stokes*, etc.

Example: Consider the *convection-diffusion problem*

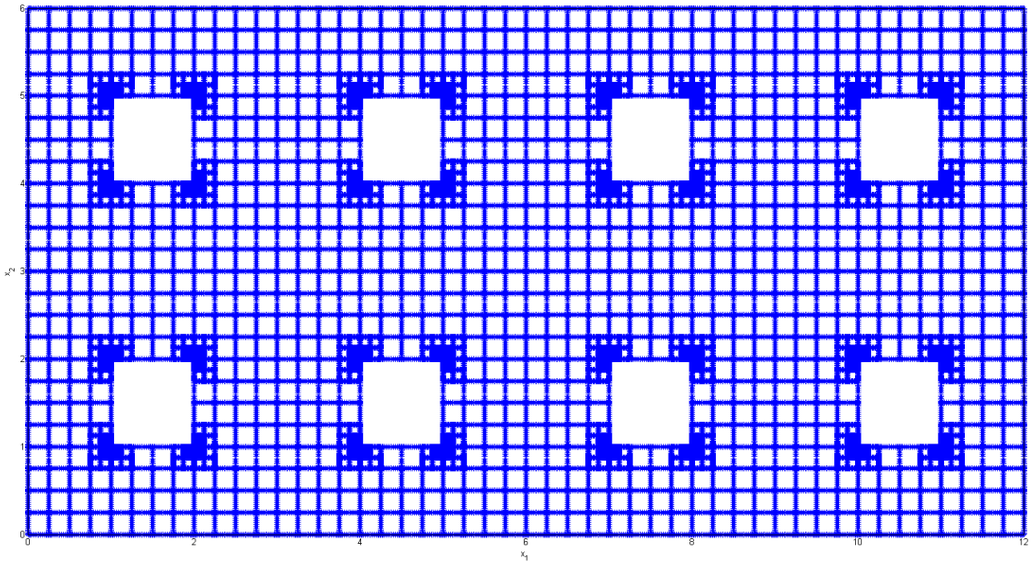
$$\frac{\partial u}{\partial t} - \Delta u + 30 \frac{\partial u}{\partial x_1} = 0,$$

defined on the domain Ω shown below:



Zero Neumann condition on blue boundaries. Periodic BC on red boundaries.

The following mesh is used (observe corner refinement!):



Observation 2: The direct solver can be used to explicitly build time-evolution operators for *hyperbolic problems.* Consider, for instance,

$$\begin{cases} \dfrac{\partial u(\boldsymbol{x},t)}{\partial t} = B\,u(\boldsymbol{x},t), & \boldsymbol{x} \in \Omega,\ t > 0 \\ u(\boldsymbol{x},0) = f(\boldsymbol{x}) & \boldsymbol{x} \in \Omega, \end{cases}$$

where B is a skew-Hermitian operator (e.g. $B = \sqrt{\Delta}$ with Dirichlet/Neumann BC). The solution is

$$u(\boldsymbol{x},t) = \big[\exp(t\,B)\,f\big](\boldsymbol{x}),$$

where $\exp(t\,B)$ is the time-evolution operator. Now suppose that we can approximate the oscillatory function $x \mapsto \exp(ix)$ by a rational function

$$R_M(ix) = \sum_{m=-M}^{M} \frac{b_m}{ix - \alpha_m},$$

where $\{b_m\}$ and $\{\alpha_m\}$ are some complex numbers such that $|R_M(ix)| \leq 1$ for $x \in \mathbb{R}$. We require that

$$\left|e^{ix} - R_M(ix)\right| \leq \delta, \qquad x \in [-\tau\Lambda, \tau\Lambda],$$

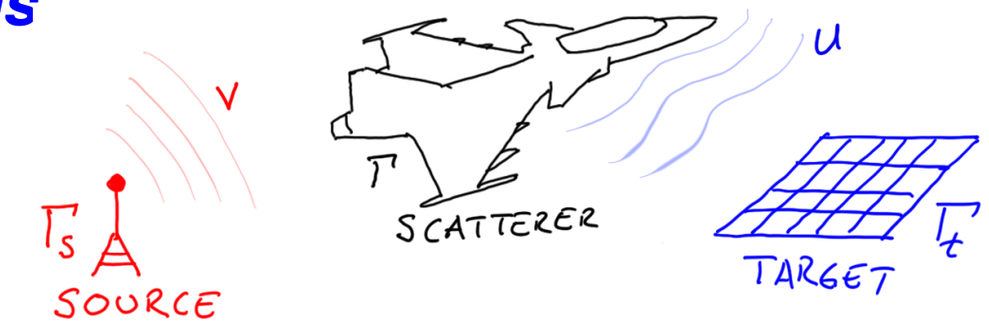
where τ is a time step, and where Λ is a "band-width" — in other words, we accurately resolve the parts of B whose spectrum fall in the interval $[-i\Lambda, i\Lambda]$. *Very high accuracy can be attained* – say $\delta = 10^{-10}$ for about 5 – 10 points per wavelength [Beylkin, Haut]. Then approximate

$$\exp(\tau B) \approx \sum_{m=-M}^{M} b_m \left(B - \alpha_m\right)^{-1}.$$

Notes: The time-step τ *can be large.* Application of $\exp(\tau B)$ is almost instantaneous. Quite high memory demands, but distributed memory is fine. ***Parallel in time!***

Current project: Shallow water equations on cubed sphere at LANL.

Part 4 (of 4) — Direct solvers for integral equations



Recall that many boundary value problems can advantageously be recast as *boundary integral equations.* Consider, e.g., (sound-soft) acoustic scattering from a finite body:

$$(6) \qquad \begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \mathbb{R}^3 \backslash \overline{\Omega} \\ u(\boldsymbol{x}) = v(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega \\ \lim\limits_{|\boldsymbol{x}|\to\infty} |\boldsymbol{x}| \left(\partial_{|\boldsymbol{x}|} u(\boldsymbol{x}) - i\kappa\, u(\boldsymbol{x}) \right) = 0. \end{cases}$$

The BVP (6) is in many ways equivalent to the BIE

$$(7) \qquad -\pi i \sigma(\boldsymbol{x}) + \int_{\partial\Omega} \left(\left(\partial_{\boldsymbol{n}(\boldsymbol{y})} + i\kappa \right) \frac{e^{i\kappa|\boldsymbol{x}-\boldsymbol{y}|}}{|\boldsymbol{x}-\boldsymbol{y}|} \right) \sigma(\boldsymbol{y})\, dS(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega.$$

The integral equation (7) has several advantages over the PDE (6), including:

- The domain of computation $\partial\Omega$ is finite.
- The domain of computation $\partial\Omega$ is 2D, while $\mathbb{R}^3 \backslash \overline{\Omega}$ is 3D.
- Equation (7) is inherently well-conditioned (as a "2^{nd} kind Fredholm equation").

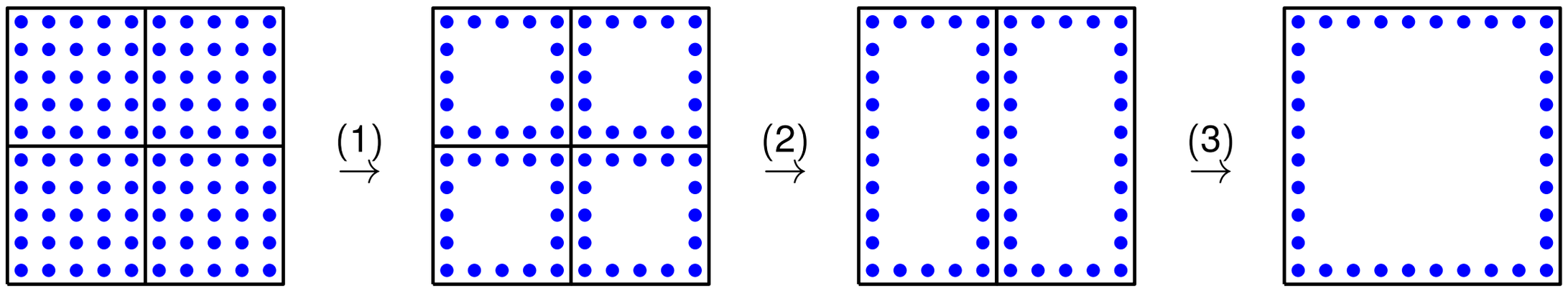
A serious drawback of integral equations is that they lead to *dense coefficient matrices.*

Since we are interested in constructing inverses anyway, this is unproblematic *for us.*

Part 4 (of 4) — Direct solvers for integral equations, continued

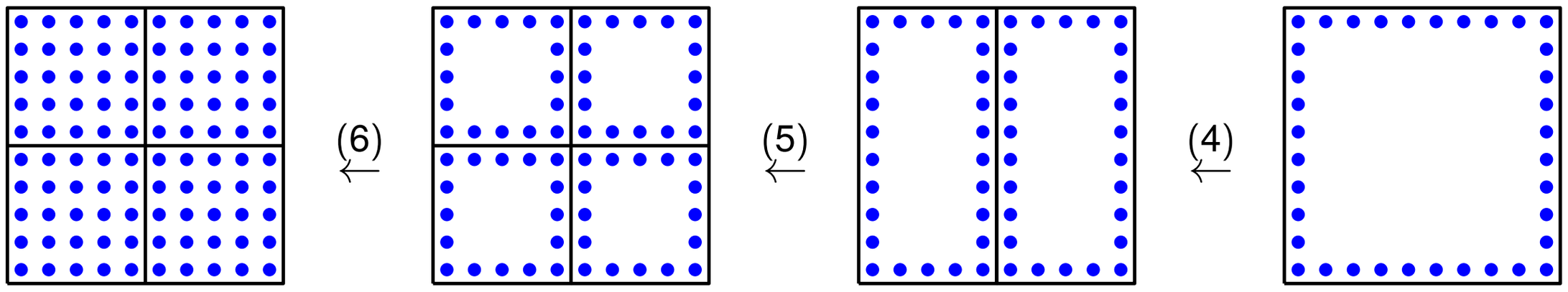
It is possible to construct direct solvers that follow the same template as before.

Upwards pass — build all solution operators:



The original grid. *Leaves reduced.* *After merge.* *After merge.*

Downwards pass — solve for a particular data function (*very* fast!):

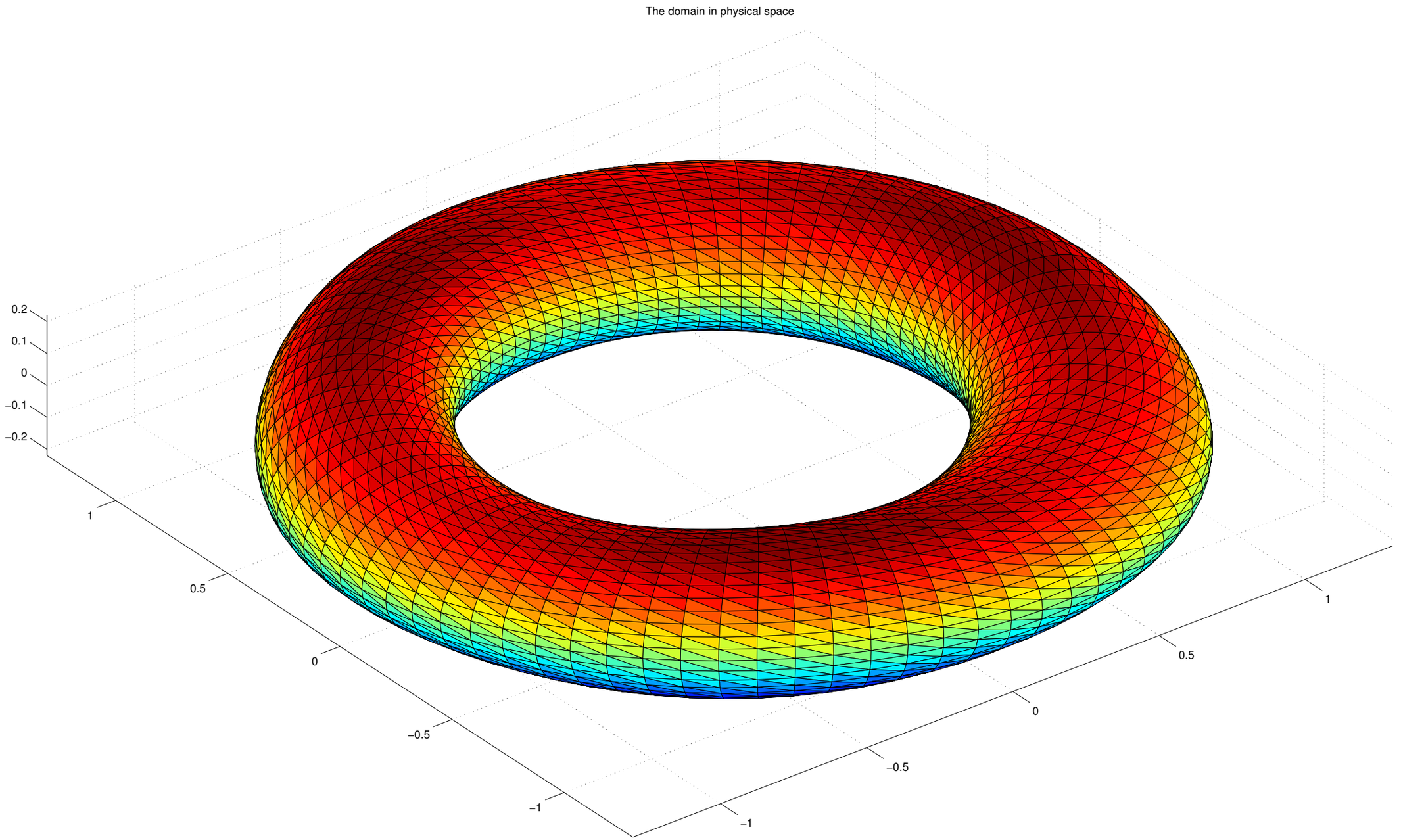


Full solution. *Solve.* *Solve.* *Top level solve.*

Our “solution operators” will be (conceptually) *scattering matrices* instead of Poincaré-Steklov operators.

The operators will no longer be pure boundary operators.

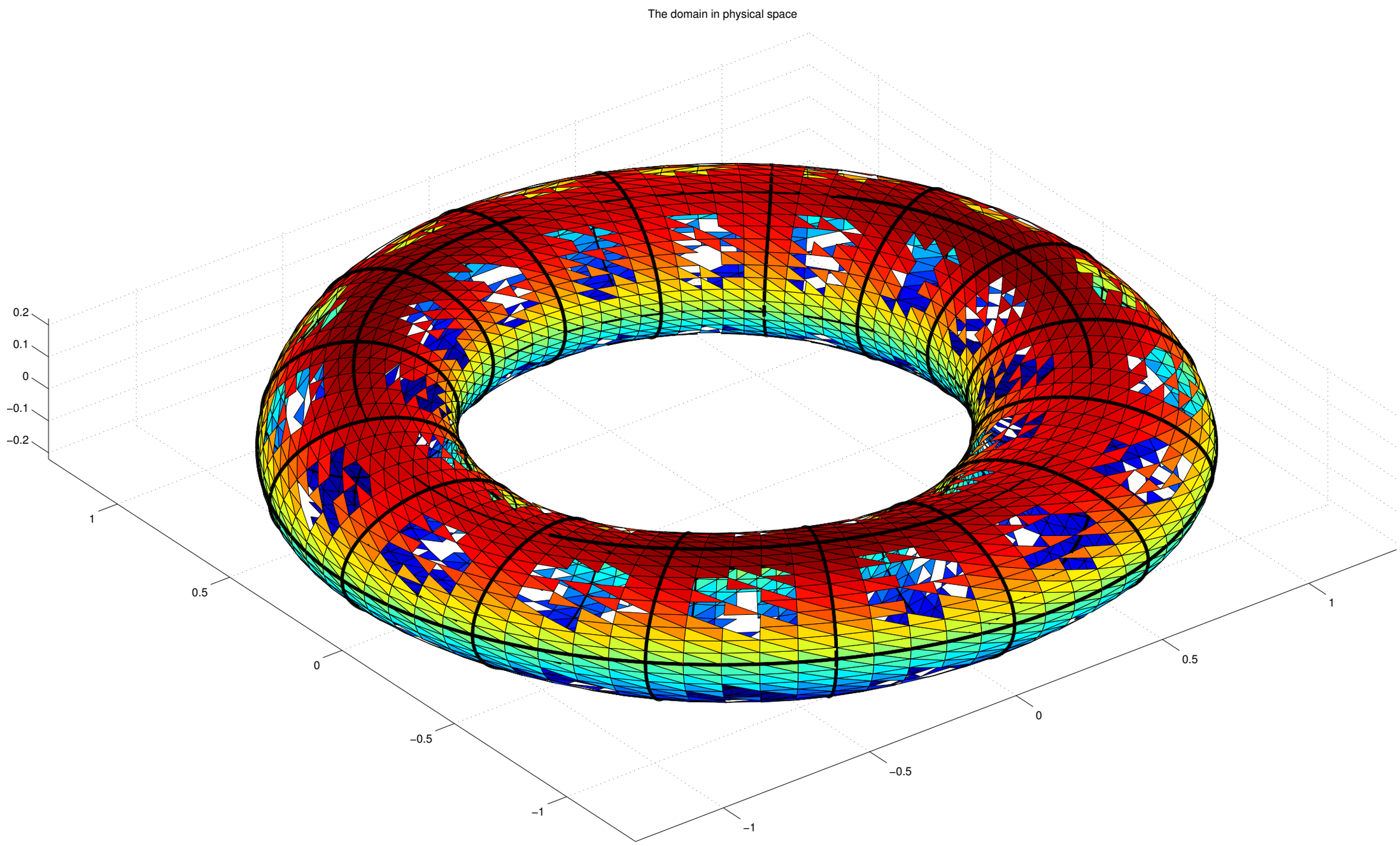
Example: BIE on a surface in $\mathbb{R}^3$:



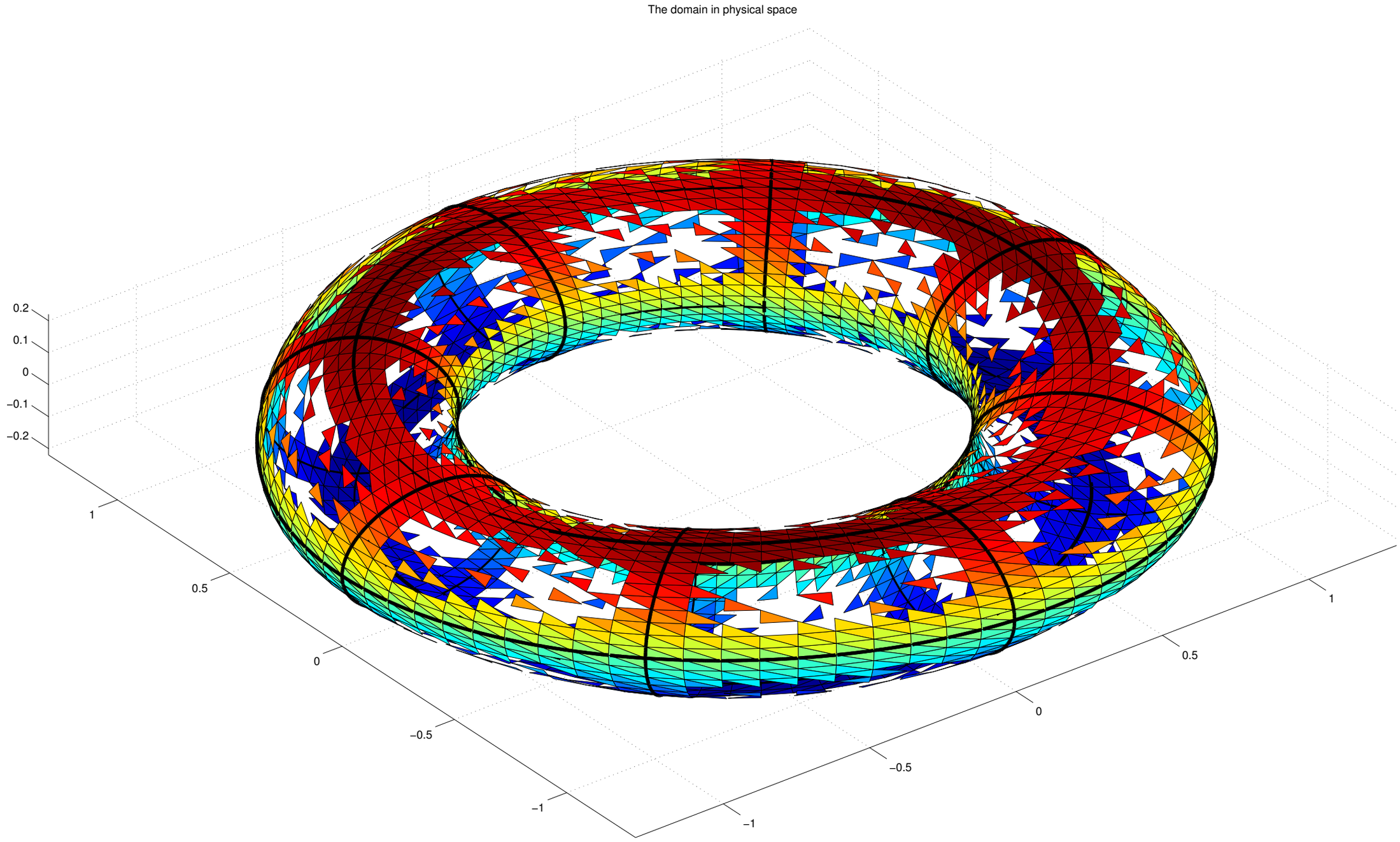
Let **A** denote an $N \times N$ matrix arising upon discretizing a boundary integral operator

$$[Aq](\boldsymbol{x}) = q(\boldsymbol{x}) + \int_\Gamma \frac{1}{|\boldsymbol{x}-\boldsymbol{y}|}\, q(\boldsymbol{y})\, dA(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Gamma,$$

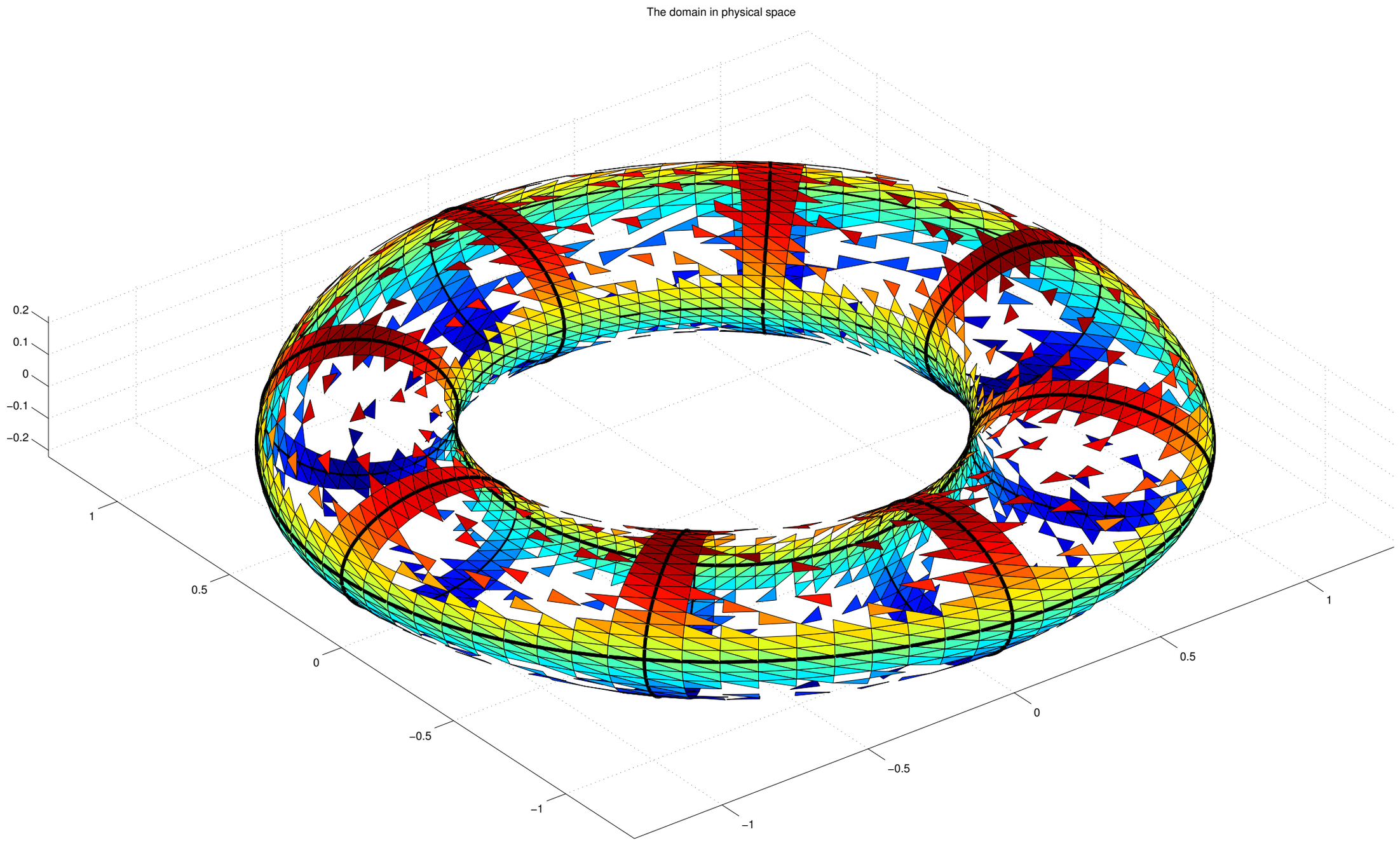
where Γ is the "torus-like" domain shown (it is deformed to avoid rotational symmetry).



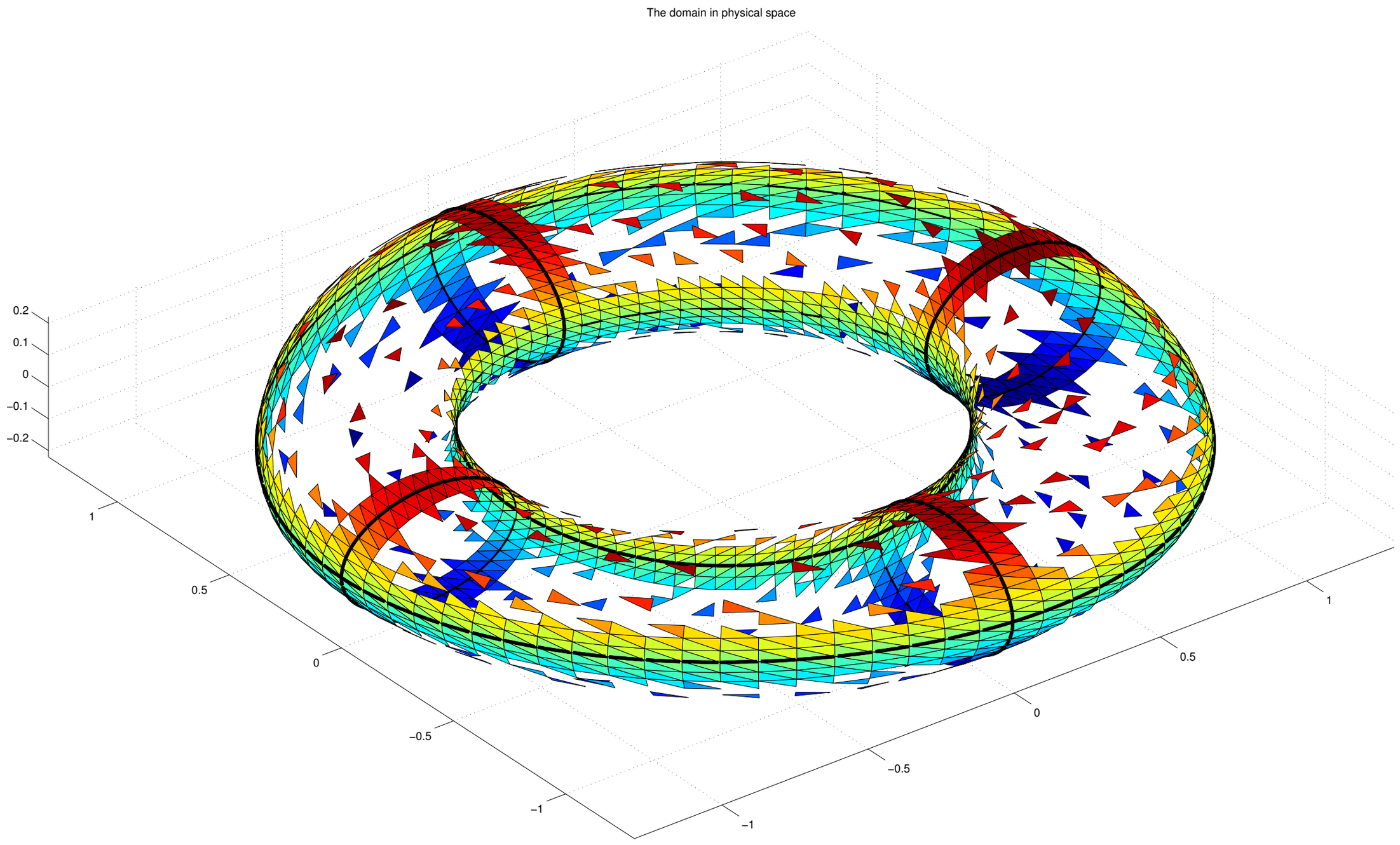
The reduced matrix represents a Nyström discretization supported on the panels shown.



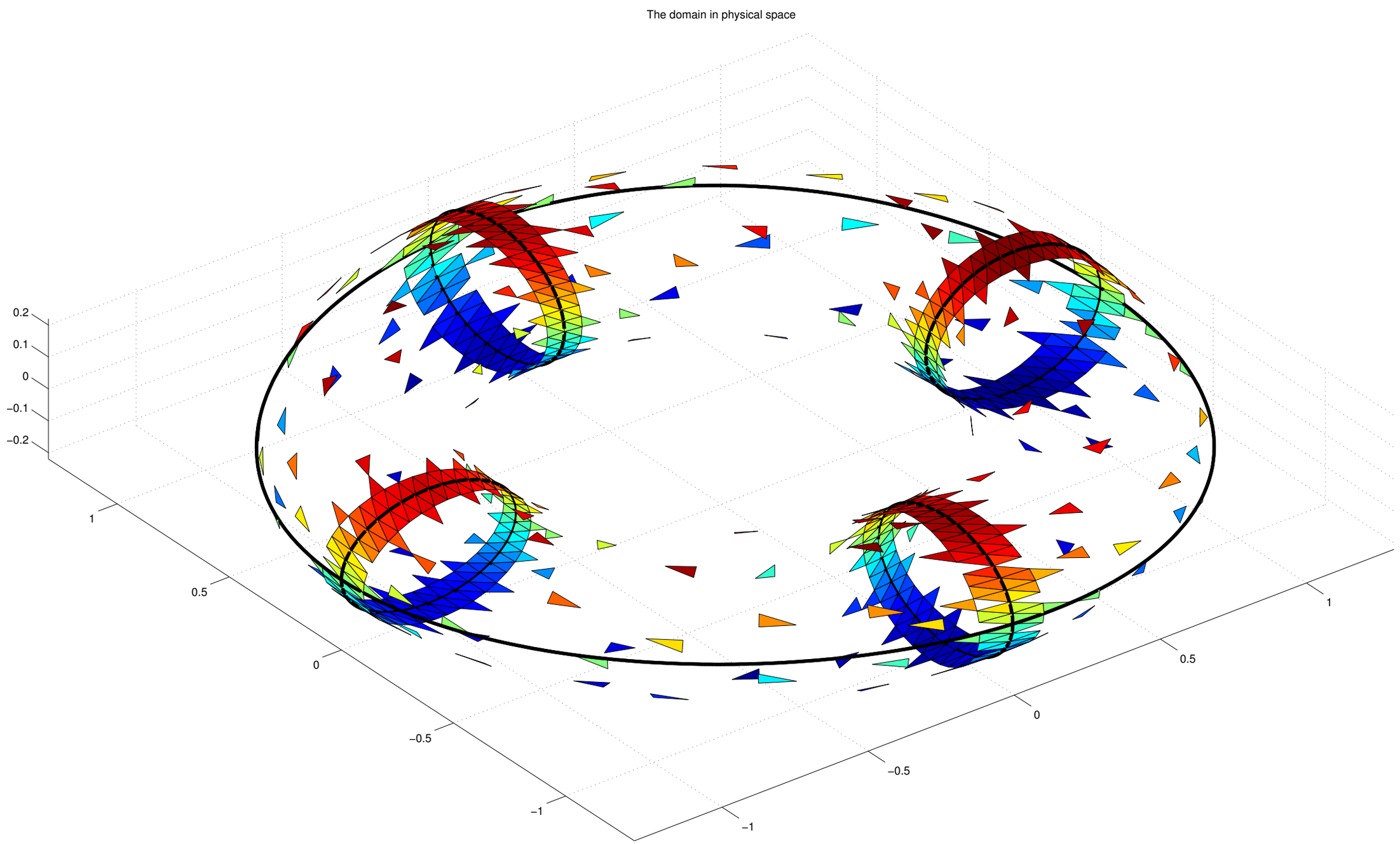
The reduced matrix represents a Nyström discretization supported on the panels shown.



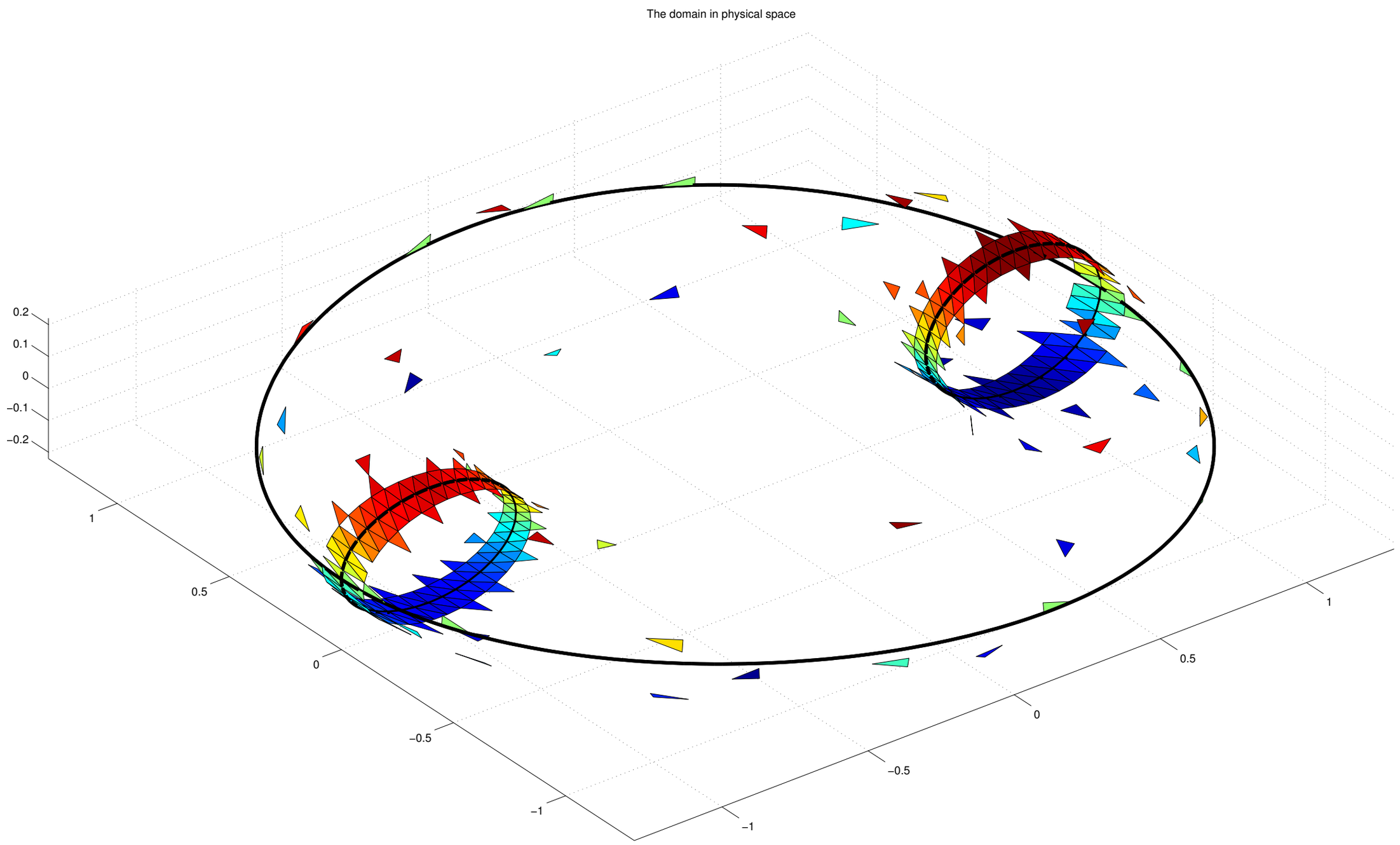
The reduced matrix represents a Nyström discretization supported on the panels shown.



The reduced matrix represents a Nyström discretization supported on the panels shown.

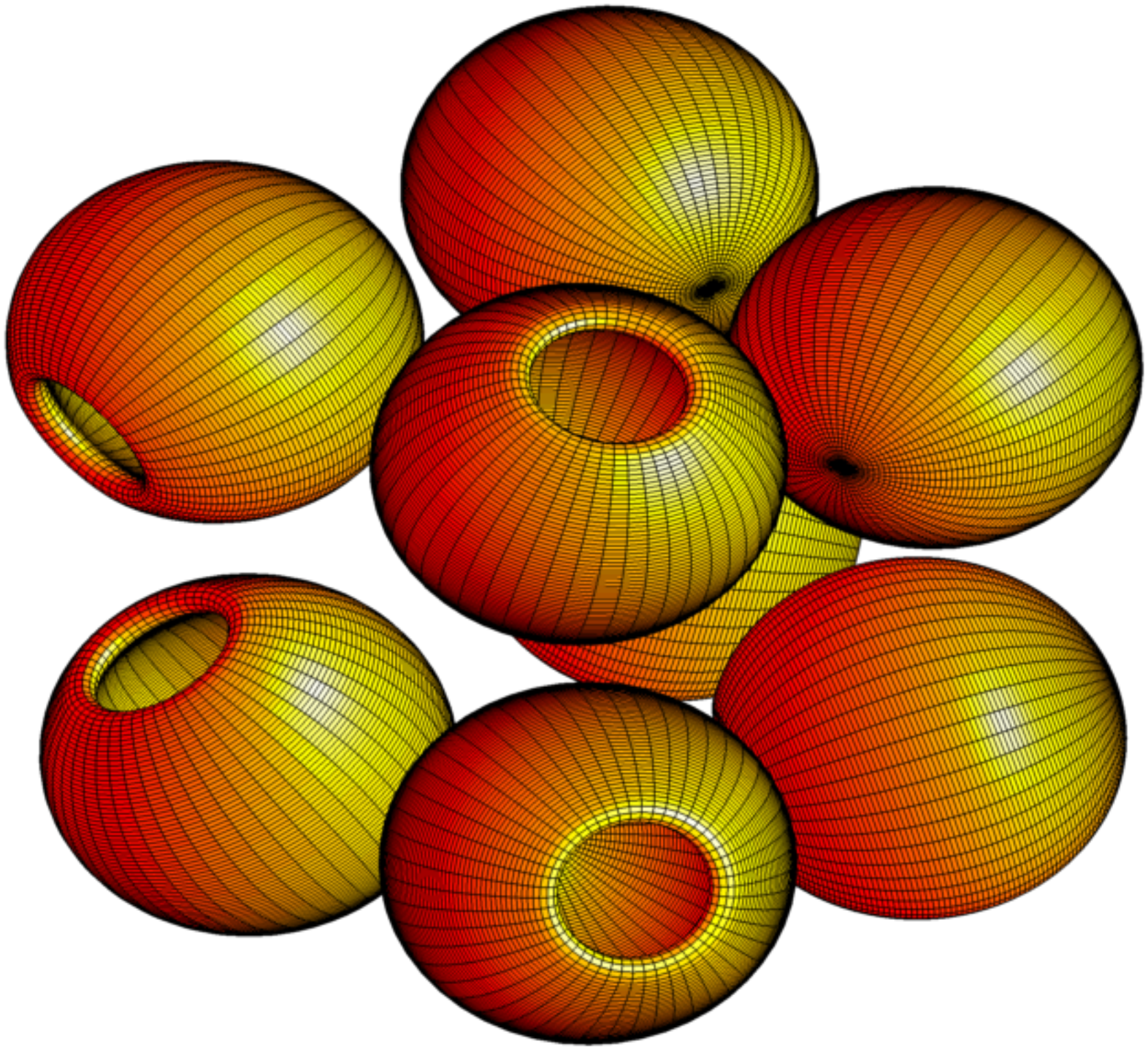


The reduced matrix represents a Nyström discretization supported on the panels shown.



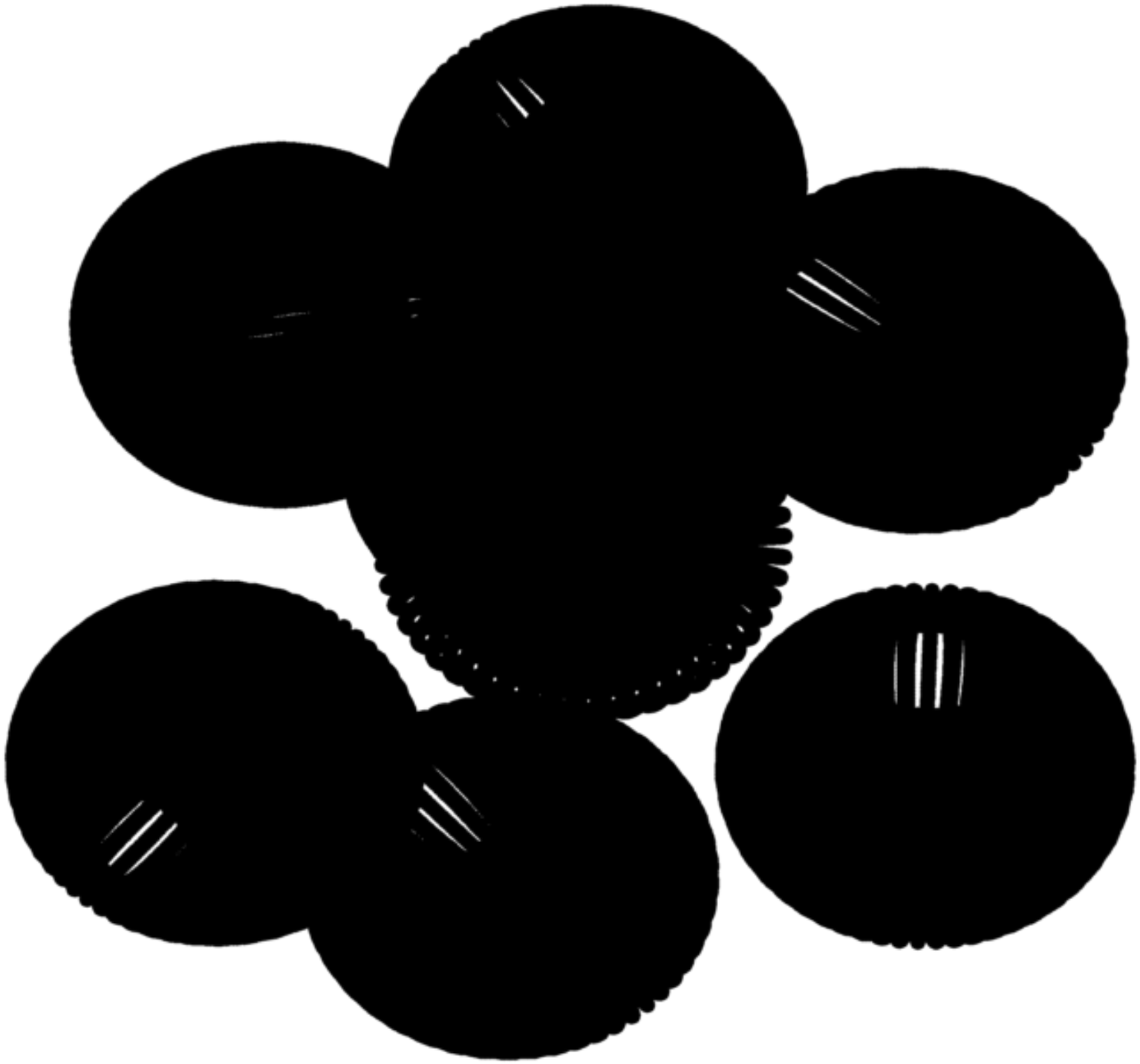
The reduced matrix represents a Nyström discretization supported on the panels shown.

Example: Multibody scattering from a domain with multiple cavities



Consider scattering from some multibody domain involving cavities.

Example: Multibody scattering from a domain with multiple cavities

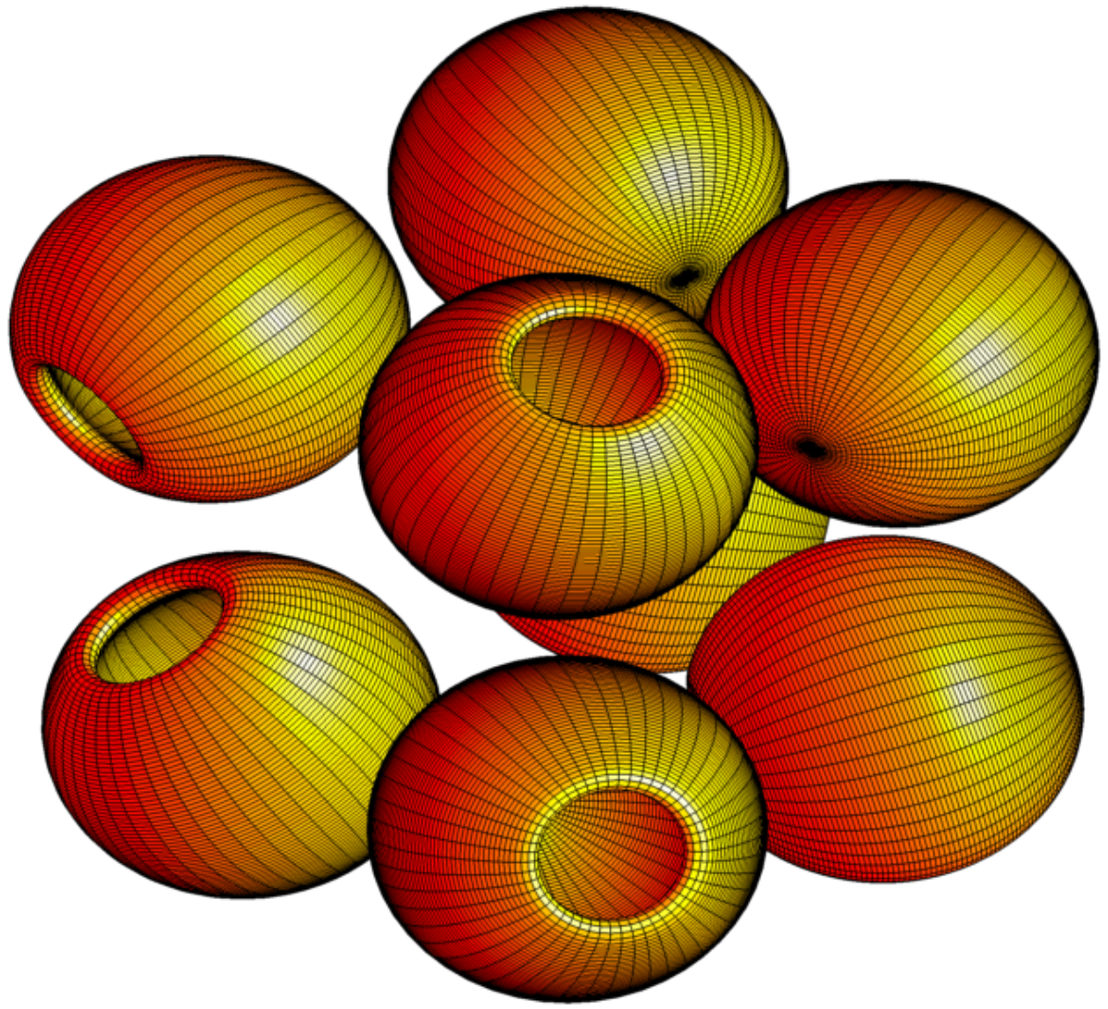


There are lots of discretization nodes involved. Very computationally intense!

Example: Multibody scattering from a domain with multiple cavities

After local compression of each scatter, the problem is much more tractable.

Example: Multibody scattering from a domain with multiple cavities



Acoustic scattering on the exterior domain.

Each bowl is about 5λ.

A hybrid direct/iterative solver is used (a highly accurate scattering matrix is computed for each body).

On an office desktop, we achieved an accuracy of 10^{-5}, in about 6h (essentially all the time is spent in applying the inter-body interactions via the Fast Multipole Method).
Accuracy 10^{-7} took 27h.

Example: BIEs on rotationally symmetric bodies (2014, with S. Hao and P. Young)

N	N_{body}	T_{fmm}	I_{GMRES} (precond /no precond)	T_{total} (precond /no precond)	E_{∞}^{rel}
10000	50× 25	1.23e+00	21 /358	2.70e+01 /4.49e+02	4.414e-04
20000	100×25	3.90e+00	21 /331	8.57e+01 /1.25e+03	4.917e-04
40000	200×25	6.81e+00	21 /197	1.62e+02 /1.18e+03	4.885e-04
80000	400×25	1.36e+01	21 / 78	3.51e+02 /1.06e+03	4.943e-04
20400	50×51	4.08e+00	21 /473	8.67e+01 /1.99e+03	1.033e-04
40800	100×51	7.20e+00	21 /442	1.56e+02 /3.17e+03	3.212e-05
81600	200×51	1.35e+01	21 /198	2.99e+02 /2.59e+03	9.460e-06
163200	400×51	2.50e+01	21 /102	5.85e+02 /2.62e+03	1.011e-05
40400	50×101	7.21e+00	21 /483	1.53e+02 /3.52e+03	1.100e-04
80800	100×101	1.34e+01	22 /452	2.99e+02 /6.31e+03	3.972e-05
161600	200×101	2.55e+01	22 /199	5.80e+02 /5.12e+03	2.330e-06
323200	400×101	5.36e+01	22 /112	1.25e+03 /5.84e+03	3.035e-06

Exterior Laplace problem solved on the multibody bowl domain with and without preconditioner.

Example: BIEs on rotationally symmetric bodies (2014, with S. Hao and P. Young)

N	N_{body}	$T_{\text{precompute}}$	I_{GMRES}	T_{solve}	E_{∞}^{rel}
80800	100×101	6.54e-01	62	5.17e+03	1.555e-03
161600	200×101	1.82e+00	63	9.88e+03	1.518e-04
323200	400×101	6.46e+00	64	2.19e+04	3.813e-04
160800	100×201	1.09e+00	63	9.95e+03	1.861e-03
321600	200×201	3.00e+00	64	2.19e+04	2.235e-05
643200	400×201	1.09e+01	64	4.11e+04	8.145e-06
641600	200×401	5.02e+00	64	4.07e+04	2.485e-05
1283200	400×401	1.98e+01	65	9.75e+04	6.884e-07

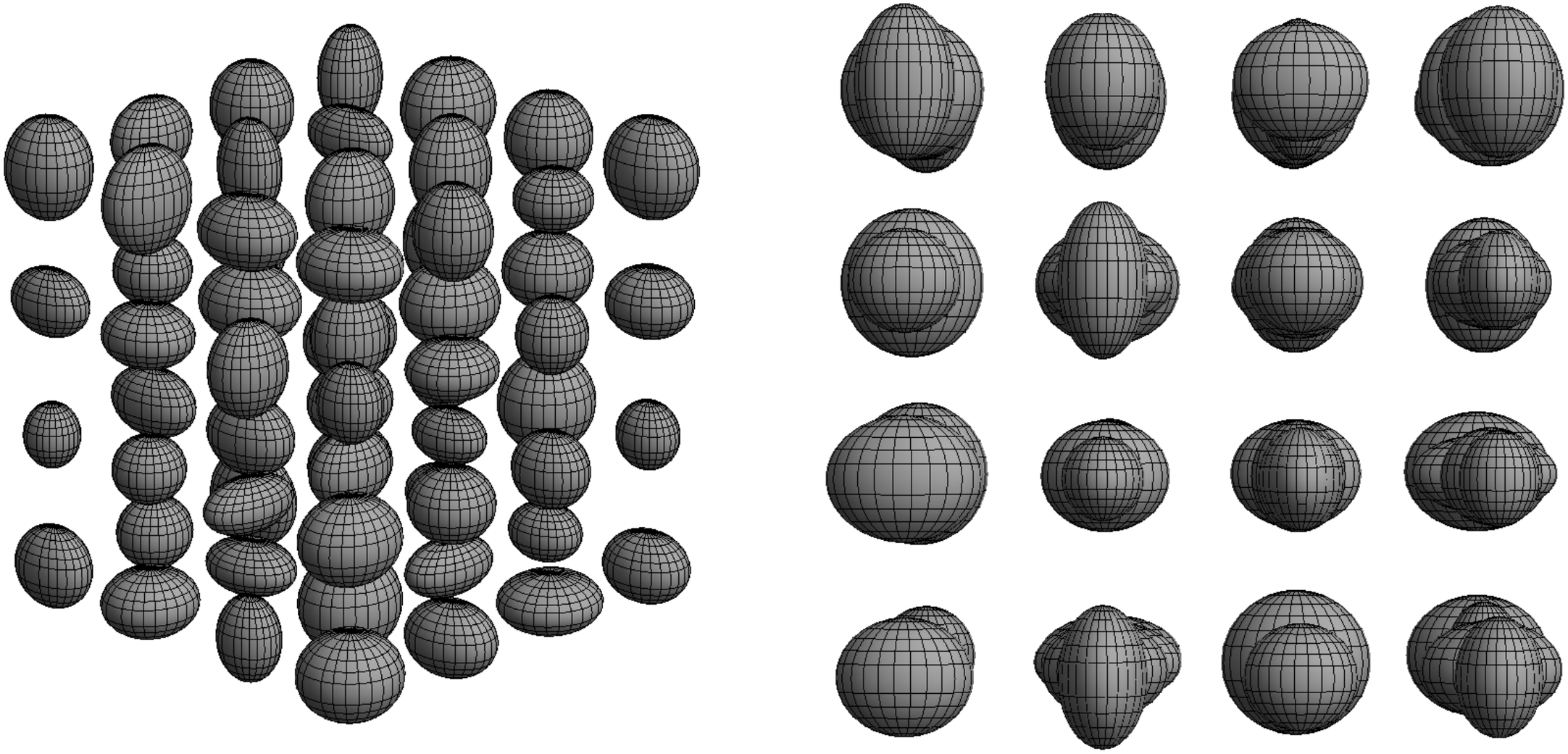
Exterior Helmholtz problem solved on multibody bowl domain.

Each bowl is 5 wavelength in diameter.

We do not give timings for standard iterative methods since in this example, they typically did not converge at all (even though the BIE is a 2nd kind Fredholm equation).

Numerical example — BIE on surfaces in 3D (2013, with J. Bremer and A. Gillman)

Consider sound-soft scattering from a multi-body scatterer of size 4 wave-lengths:



The global scattering matrix is computed using the hierarchical direct solver described.

(The ellipsoids are not rotationally symmetric.)

Numerical example — BIE on surfaces in 3D (2013, with J. Bremer and A. Gillman)

The local truncation error is set to 10^{-3}.

Grid dimensions	N	T	E	Ratio	Predicted
$2 \times 2 \times 2$	12 288	$1.02 \times 10^{+1}$	3.37×10^{-04}	-	-
$3 \times 3 \times 3$	41 472	$3.43 \times 10^{+1}$	4.81×10^{-04}	3.4	6.2
$4 \times 4 \times 4$	98 304	$7.92 \times 10^{+1}$	1.57×10^{-04}	2.3	3.7
$6 \times 6 \times 6$	331 776	$2.96 \times 10^{+2}$	7.03×10^{-04}	3.7	6.2
$8 \times 8 \times 8$	786 432	$6.70 \times 10^{+2}$	4.70×10^{-04}	2.3	3.7
$10 \times 10 \times 10$	1 536 000	$2.46 \times 10^{+3}$	3.53×10^{-04}	3.7	2.7

Increasing the accuracy is possible, but comes at a cost.

Now the local truncation error is set to 10^{-6}.

Grid dimensions	N	T	E	Ratio	Predicted
$2 \times 2 \times 2$	49 152	$1.61 \times 10^{+2}$	1.22×10^{-07}	-	-
$3 \times 3 \times 3$	165 888	$6.87 \times 10^{+2}$	4.92×10^{-07}	4.3	6.2
$4 \times 4 \times 4$	393 216	$1.68 \times 10^{+3}$	5.31×10^{-07}	2.4	3.6
$6 \times 6 \times 6$	1 327 104	$6.66 \times 10^{+3}$	4.60×10^{-06}	4.0	6.2
$8 \times 8 \times 8$	3 145 728	$1.59 \times 10^{+4}$	2.30×10^{-07}	2.4	3.6

Example: Acoustic scattering from a “deformed torus” (with J. Bremer and A. Gillman)

The domain is roughly $2 \times 2 \times 0.7$ *wave-lengths in size.*

$N_{\text{triangles}}$	N	T	E
32	1 664	$7.16 \times 10^{+00}$	3.51×10^{-02}
128	6 656	$6.29 \times 10^{+01}$	4.41×10^{-03}
512	26 624	$2.81 \times 10^{+02}$	4.08×10^{-05}
2 048	106 496	$2.60 \times 10^{+03}$	7.80×10^{-07}
8 192	425 984	$1.47 \times 10^{+04}$	3.25×10^{-08}

(Note: Laplace problems are much faster.)

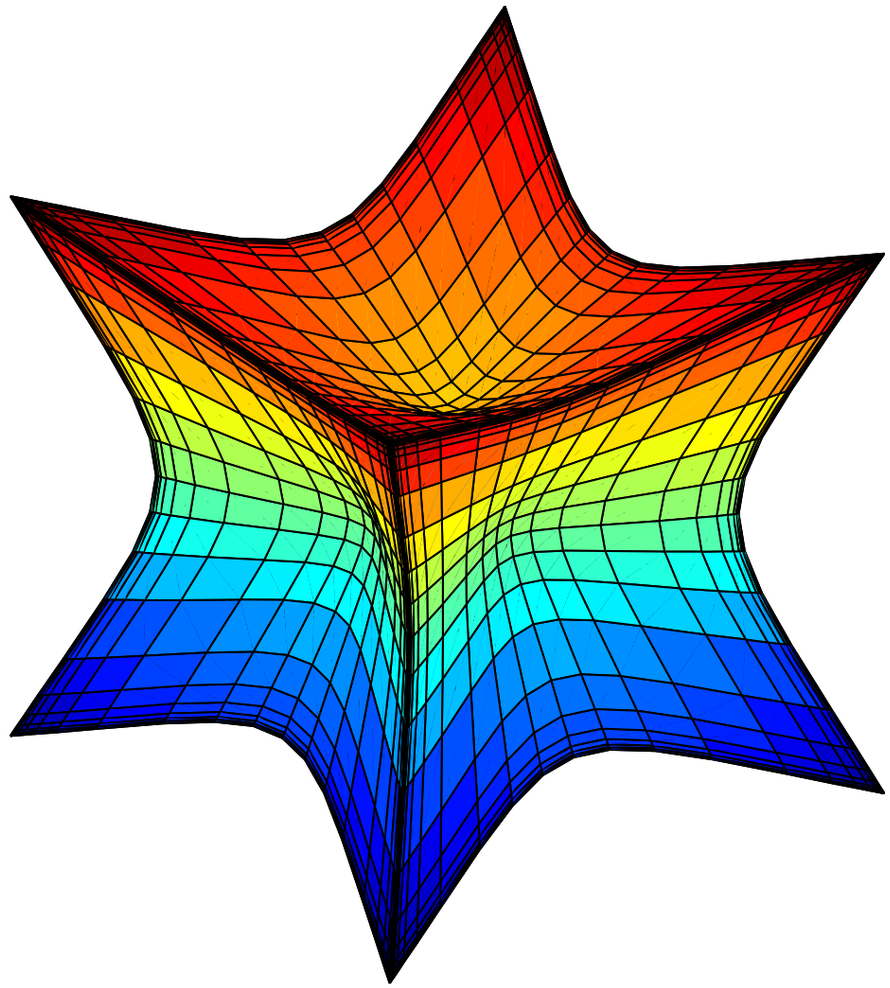
Numerical example — BIE on "edgy" surface (2013, with J. Bremer and A. Gillman)

A surface Γ with corners and edges.

The grid has been refined to attain high accuracy.

Computing scattering matrices for the corners is conceptually easy (but laborious). The direct solver eliminates "extra" DOFs.

Compressing the edges takes effort!



N_{tris}	N	E	T	$N_{\text{out}} \times N_{\text{in}}$
192	21 504	2.60×10^{-08}	$6.11 \times 10^{+02}$	617×712
432	48 384	2.13×10^{-09}	$1.65 \times 10^{+03}$	620×694
768	86 016	3.13×10^{-10}	$3.58 \times 10^{+03}$	612×685

Results from a Helmholtz problem (acoustic scattering) on the domain exterior to the "edgy" cube.

The domain is about 3.5 wave-lengths in diameter.

Note: We compress patches that are *directly adjacent.*

This is in contrast to, e.g., the Fast Multipole Methods, $\mathcal{H}$- and $\mathcal{H}^2$-matrix methods, etc.

Advantages: Easier data structures, more efficient inversion, better localization of data (leading to algorithms that are easier to parallelize).

Disadvantages: Ranks are higher, sometimes much higher.
Numerical compression is required.

Additional machinery required to attain $O(N)$ complexity in 3D:

- Use Nested hierarchies — the dense blocks themselves have structure.
  - E. Corona, P.G. Martinsson, D. Zorin *"An $O(N)$ Direct Solver for Integral Equations in the Plane"* *Advances in Computational and Harmonic Analysis,* **38**(2), 2015, pp. 284–317.
- Use multiple, staggered, grids.
  - K. Ho and L. Ying, *"Hierarchical interpolative factorization for elliptic operators: differential equations."* Communications on Pure and Applied Mathematics (2015).
  - K. Ho and L. Ying, *"Hierarchical interpolative factorization for elliptic operators: integral equations."* Communications on Pure and Applied Mathematics (2015).

Numerical example — Volume int. eq. in 2D (2013, with E. Corona and D. Zorin)

Consider a volume integral equation in the plane:

$$q(x) + \int_\Omega b(x) \log|x - y|\, q(y)\, dy = f(x), \qquad x \in \Omega,$$

where $\Omega = [0, 1]^2$, and where

$$b(x) = 1 + 0.5e^{-(x_1-0.3)^2-(x_2-0.6)^2}.$$

The domain is discretized on a uniform grid, with simplistic quadrature.

By exploiting internal structure (HBS structure) in the scattering matrices, we have built a direct solver with optimal $O(N)$ complexity for every step.

Numerical example — Volume int. eq. in 2D (2013, with E. Corona and D. Zorin)

N	T_{build}	T_{solve}	Memory	Error
784	0.17 s	0.002 s	4.48 MB	1.6e-14
3,136	1.70 s	0.009 s	25.24 MB	1.8e-14
12,544	8.32 s	0.036 s	123.07 MB	8.6e-11
50,176	40.43 s	0.155 s	538.51 MB	1.6e-10
200,704	3.23 m	0.677 s	2.23 GB	2.3e-10
802,816	13.66 m	2.819 s	9.23 GB	4.0e-10
3,211,264	54.79 m	11.737 s	34.09 GB	5.1e-09

Execution times in Matlab, on an Intel Xeon X5650 (6 core) 2.67 GHz.

For a computed approximate inverse $\mathbf{B} \approx \mathbf{A}^{-1}$, the error reported is

$$\text{Error} = \max_i \frac{\|\mathbf{v}^{(i)} - \mathbf{AB}\mathbf{v}^{(i)}\|}{\|\mathbf{v}^{(i)}\|}$$

where $\{\mathbf{v}^{(i)}\}_{i=1}^{10}$ is a collection of random vectors.

Key points:

- Direct solvers with $O(N)$ complexity exist for elliptic PDEs with non-oscillatory (or "mildly oscillatory") solutions for most standard environments:
  - Sparse matrices from FEM/FD/composite spectral/... in both 2D and 3D.
  - Boundary integral equations in 2D and 3D.
  - Practical efficiency of "build stage" in 3D requires further work.

- Often *instantaneous solves* once a solution operator has been constructed.
  - Scattering problems, engineering design, optimization, ...
  - Time-stepping certain evolution problems.

- Can often *eliminate problems with slow convergence of iterative solvers.*

- Direct solvers play very nicely with *high-order discretizations.*
  Such discretizations ameliorate high memory costs of direct solvers.

- Direct solvers are excellent for locally refined meshes — *corners, edges, etc.*

- *Randomized methods for compressing matrices* are extremely useful in this context.

Future directions: *New areas of research → much work remains!*

- Much remains to do with randomized methods in linear algebra: (1) Improved algorithms for computing rank-revealing factorizations. (2) Compression of sparse and structured matrices. (3) Finding improved random projections that allow fast application to various types of matrices. (4) *[High risk/high reward]* Accelerate linear solvers for general systems $\mathbf{Ax} = \mathbf{b}$. The goal is complexity $O(N^{2+\epsilon})$ for small ϵ, while retaining stability, and high practical efficiency. Et cetera.
- For the direct solvers, *mathematical theory* is sorely lacking. "Signing the blueprint." Efficient and accurate à posteriori error estimation techniques do exist.

  Current discretizations based on "hard" constraints — pointwise collocation, etc.

  To develop theory, very likely advantageous to switch to variational formulations.
- *Exploit communication efficiency of direct solvers!*
  - Low-hanging fruit for mid-size problems where the structured matrix algebra on matrices of size $O(N^{2/3}) \times O(N^{2/3})$ can be done on one node.
  - In the medium term, hybrid solvers that use direct solves for subproblems (e.g. near refined points in a mesh) will be highly useful.
  - The development of fully parallel codes where every component is executed using distributed memory will require hard work. Potentially huge rewards, however.
- Extension to parabolic and hyperbolic equations. *Parallel in time.*

To find out more:

- Papers, talk slides, some codes:

  `http://amath.colorado.edu/faculty/martinss/`

  (Google “Gunnar Martinsson”)

- Summer school (CBMS) on direct solvers at Dartmouth College, June, 2014.

  Slides, tutorial codes, etc, at:

  `http://amath.colorado.edu/faculty/martinss/2014_CBMS/`

  `http://www.math.dartmouth.edu/~fastdirect/`

  Videos of all 10 lectures are available — google “Fast Direct Solvers youtube.”

- Summer school on randomized methods in numerical linear algebra:
  IAS / Park City Mathematics Institute, July 2016