

Direct solvers for elliptic PDEs

Gunnar Martinsson

Dept. of Mathematics & Oden Institute for Computational Sciences and Engineering

University of Texas at Austin

Students, postdocs, collaborators: Daniel Appelö, Chao Chen, Ke Chen, Yijun Dong, Adrianna Gillman, Abinand Gopal, James Levitt, Bowei Wu, Anna Yesypenko.

Slides: http://users.oden.utexas.edu/~pgm/main_talks.html

Research support by:



Problem addressed: The talk concerns numerical methods for boundary value problems of the form

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

where Ω is a domain with boundary Γ , and where A is a linear elliptic differential operator (with possibly variable coefficients).

Examples of problems we are interested in:

- 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).

Archetypical example: Poisson equation with Dirichlet boundary data:

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

Standard numerical recipe: (1) Discretize. (2) Solve sparse linear system iteratively.

Problem addressed: The talk concerns numerical methods for boundary value problems of the form

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

where Ω is a domain with boundary Γ , and where A is a linear elliptic differential operator (with possibly variable coefficients).

Observation: The problem is *in principle* easy to solve! Consider the Poisson equation

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

(with suitable decay conditions at infinity to ensure uniqueness).

Problem addressed: The talk concerns numerical methods for boundary value problems of the form

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

where Ω is a domain with boundary Γ , and where A is a linear elliptic differential operator (with possibly variable coefficients).

Observation: The problem is *in principle* easy to solve! Consider the Poisson equation

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

(with suitable decay conditions at infinity to ensure uniqueness). The solution is given by

$$(SLN) \quad u(\mathbf{x}) = \int_{\mathbb{R}^2} \phi(\mathbf{x} - \mathbf{y}) g(\mathbf{y}) d\mathbf{y}, \quad \mathbf{x} \in \mathbb{R}^2.$$

where the “fundamental solution” of the Laplace operator $-\Delta$ on \mathbb{R}^2 is defined by

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

(Evaluating (SLN) numerically is harder than it looks — more on that shortly.)

Problem addressed: The talk concerns numerical methods for boundary value problems of the form

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

where Ω is a domain with boundary Γ , and where A is a linear elliptic differential operator (with possibly variable coefficients).

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

Problem addressed: The talk concerns numerical methods for boundary value problems of the form

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

where Ω is a domain with boundary Γ , and where A is a linear elliptic differential operator (with possibly variable coefficients).

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

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

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

Good: The operators in (SLN) are friendly and nice.

Bounded, smoothing, often fairly stable, etc.

Problem addressed: The talk concerns numerical methods for boundary value problems of the form

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

where Ω is a domain with boundary Γ , and where A is a linear elliptic differential operator (with possibly variable coefficients).

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

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

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

Good: The operators in (SLN) are friendly and nice.

Bounded, smoothing, often fairly stable, etc.

Bad: The kernels G and F in (SLN) are generally *unknown*.

(Other than in trivial cases — constant coefficients and very simple domains.)

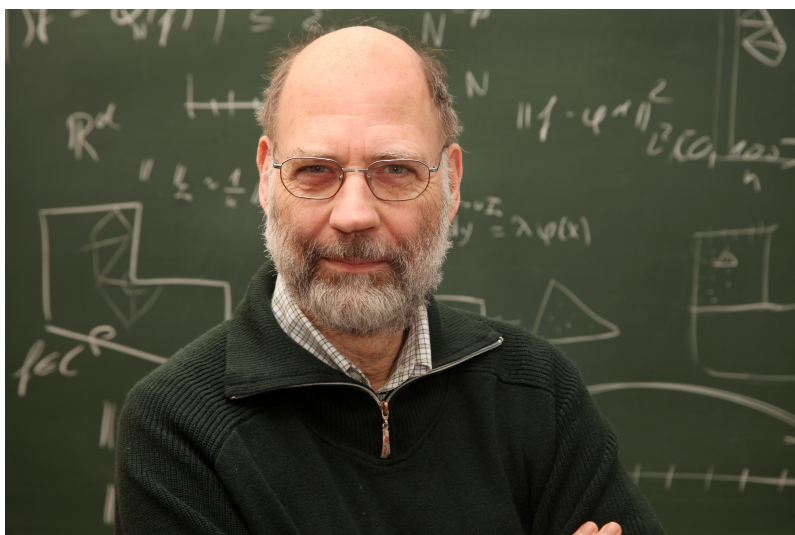
Bad: The operators in (SLN) are *global*.

Dense matrices upon discretization. $O(N^2)$ cost? $O(N^3)$ cost?



Greengard, Rokhlin (1985): The solution operator for the Poisson equation can be applied in $O(N)$ operations. “Fast Multipole Method.”

Beylkin, Coifman, Rokhlin (1991): Fast algorithms exist for most solution operators.



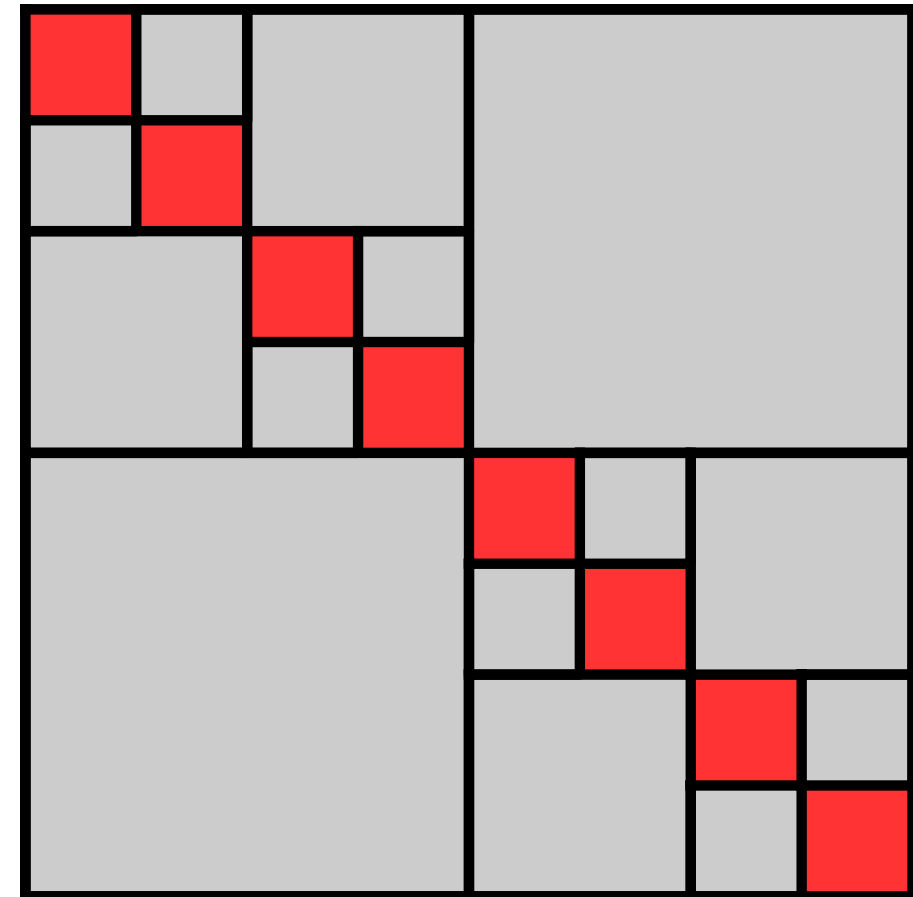
Hackbusch et al (1998): Explicit recipe for building the operators in $O(n \log^r n)$ operations for r moderate. “ \mathcal{H} -matrices.”

Recall that we are interested in solving the PDE
$$\begin{cases} Au(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ Bu(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases} \quad (\text{BVP})$$

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

Recurring idea: Upon discretization, (SLN) leads to a matrix with *off-diagonal blocks of low numerical rank*.

This property can be exploited to attain linear or close to linear complexity for operations such as matrix-vector multiply, matrix-matrix multiply, LU factorization, matrix inversion, forming of Schur complements, etc.



All gray blocks have low rank.

Strong connections to Calderón-Zygmund theory for singular integral operators.

References: Fast Multipole Method (Greengard, Rokhlin); Panel Clustering (Hackbusch); \mathcal{H} - and \mathcal{H} -matrices; Hierarchically Block Separable (HBS) matrices; Hierarchically Semi Separable (HSS) matrices; S-matrices, a.k.a. HODLR matrices;...

Example: We seek to invert a matrix \mathbf{A} as shown. Each block is of size $n \times n$, and \mathbf{A}_{12} and \mathbf{A}_{21} have rank $k < n$.

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

Example: We seek to invert a matrix \mathbf{A} as shown. Each block is of size $n \times n$, and \mathbf{A}_{12} and \mathbf{A}_{21} have rank $k < n$.

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

We first form low-rank factorizations of \mathbf{A}_{12} and \mathbf{A}_{21} so that

$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{B}_{12} \mathbf{V}_2^* \quad \text{and} \quad \mathbf{A}_{21} = \mathbf{U}_2 \mathbf{B}_{21} \mathbf{V}_1^*$$

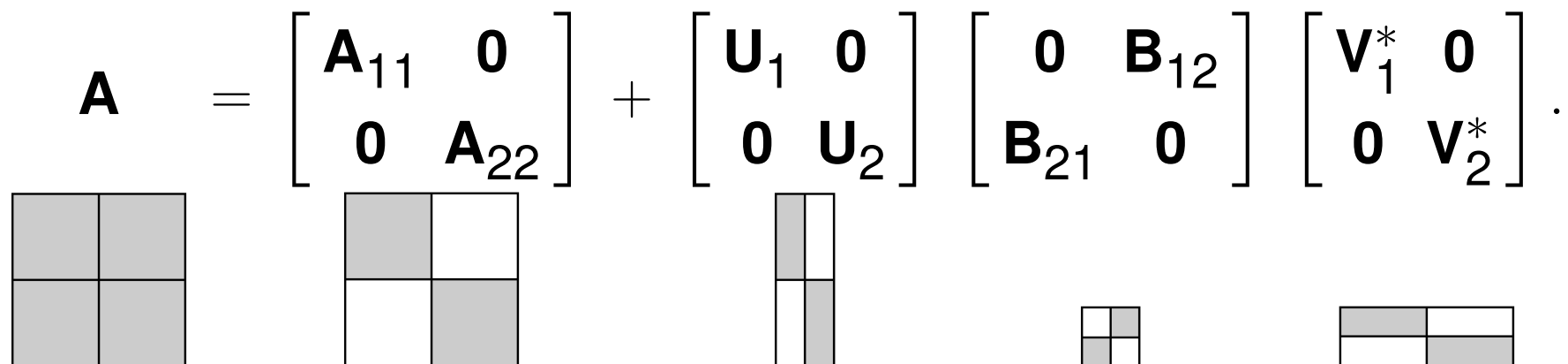

Example: We seek to invert a matrix \mathbf{A} as shown. Each block is of size $n \times n$, and \mathbf{A}_{12} and \mathbf{A}_{21} have rank $k < n$.

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

We first form low-rank factorizations of \mathbf{A}_{12} and \mathbf{A}_{21} so that

$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{B}_{12} \mathbf{V}_2^* \quad \text{and} \quad \mathbf{A}_{21} = \mathbf{U}_2 \mathbf{B}_{21} \mathbf{V}_1^*$$

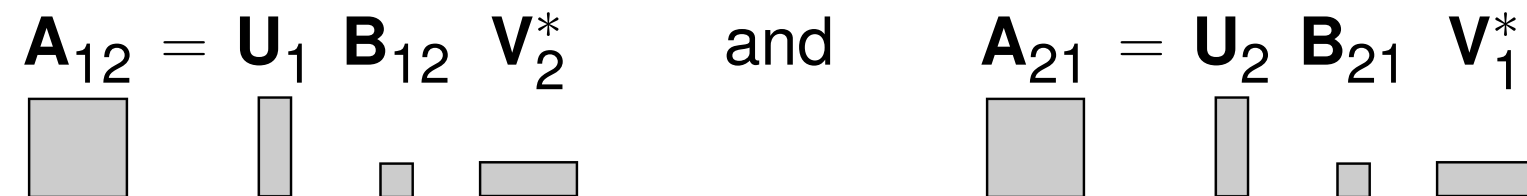

Then we can write \mathbf{A} in the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \end{bmatrix}.$$


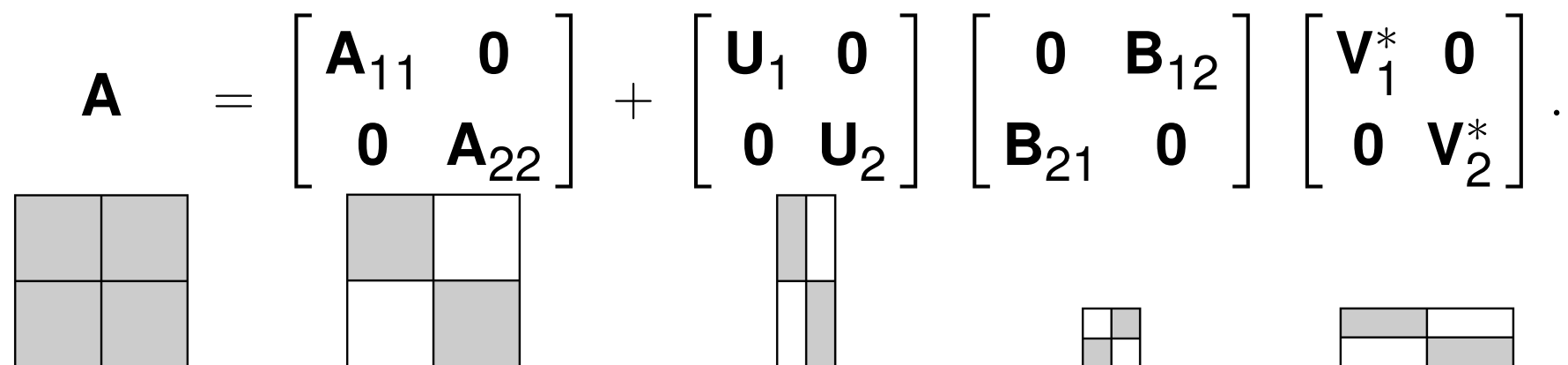
Example: We seek to invert a matrix \mathbf{A} as shown. Each block is of size $n \times n$, and \mathbf{A}_{12} and \mathbf{A}_{21} have rank $k < n$.

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

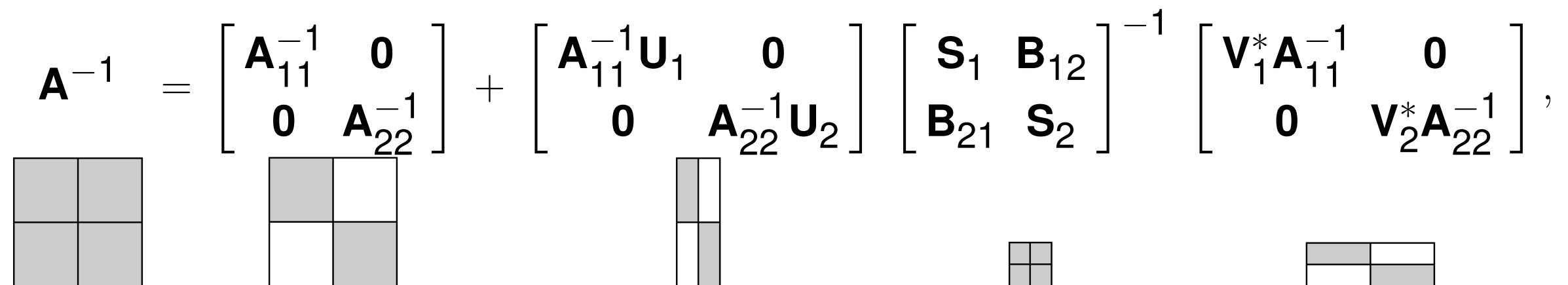
We first form low-rank factorizations of \mathbf{A}_{12} and \mathbf{A}_{21} so that

$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{B}_{12} \mathbf{V}_2^* \quad \text{and} \quad \mathbf{A}_{21} = \mathbf{U}_2 \mathbf{B}_{21} \mathbf{V}_1^*$$


Then we can write \mathbf{A} in the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \end{bmatrix}.$$


Applying the Woodbury formula, we get

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{11}^{-1} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{S}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{V}_1^* \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \mathbf{A}_{22}^{-1} \end{bmatrix},$$


where $\mathbf{S}_1 = (\mathbf{V}_1^* \mathbf{A}_{11}^{-1} \mathbf{U}_1)^{-1}$ and $\mathbf{S}_2 = (\mathbf{V}_2^* \mathbf{A}_{22}^{-1} \mathbf{U}_2)^{-1}$.

Example: We seek to invert a matrix \mathbf{A} as shown. Each block is of size $n \times n$, and \mathbf{A}_{12} and \mathbf{A}_{21} have rank $k < n$.

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

We first form low-rank factorizations of \mathbf{A}_{12} and \mathbf{A}_{21} so that

$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{B}_{12} \mathbf{V}_2^* \quad \text{and} \quad \mathbf{A}_{21} = \mathbf{U}_2 \mathbf{B}_{21} \mathbf{V}_1^*$$

Then we can write \mathbf{A} in the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \end{bmatrix}.$$

Applying the Woodbury formula, we get

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{11}^{-1} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{S}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{V}_1^* \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \mathbf{A}_{22}^{-1} \end{bmatrix},$$

$2n \times 2n$ $2n \times 2n$ $2n \times 2k$ $2k \times 2k$ $2k \times 2n$

where $\mathbf{S}_1 = (\mathbf{V}_1^* \mathbf{A}_{11}^{-1} \mathbf{U}_1)^{-1}$ and $\mathbf{S}_2 = (\mathbf{V}_2^* \mathbf{A}_{22}^{-1} \mathbf{U}_2)^{-1}$.

Example: We seek to invert a matrix \mathbf{A} as shown. Each block is of size $n \times n$, and \mathbf{A}_{12} and \mathbf{A}_{21} have rank $k < n$.

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

We first form low-rank factorizations of \mathbf{A}_{12} and \mathbf{A}_{21} so that

$$\mathbf{A}_{12} = \mathbf{U}_1 \mathbf{B}_{12} \mathbf{V}_2^* \quad \text{and} \quad \mathbf{A}_{21} = \mathbf{U}_2 \mathbf{B}_{21} \mathbf{V}_1^*$$

Then we can write \mathbf{A} in the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^* & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \end{bmatrix}.$$

Applying the Woodbury formula, we get

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{11}^{-1} \mathbf{U}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{S}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{V}_1^* \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* \mathbf{A}_{22}^{-1} \end{bmatrix},$$

$2n \times 2n$ $2n \times 2n$ $2n \times 2k$ $2k \times 2k$ $2k \times 2n$

where $\mathbf{S}_1 = (\mathbf{V}_1^* \mathbf{A}_{11}^{-1} \mathbf{U}_1)^{-1}$ and $\mathbf{S}_2 = (\mathbf{V}_2^* \mathbf{A}_{22}^{-1} \mathbf{U}_2)^{-1}$. So to get \mathbf{A}^{-1} , we need to:

- Compute \mathbf{A}_{11}^{-1} and \mathbf{A}_{22}^{-1} . *Two inverses of half the size.*
- Form \mathbf{S}_1 and \mathbf{S}_2 , and then invert $\begin{bmatrix} \mathbf{S}_1 & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{S}_2 \end{bmatrix}$. *This is a small ($2k \times 2k$) matrix.*
- Form various matrix-matrix products involving at least one “thin” matrix.

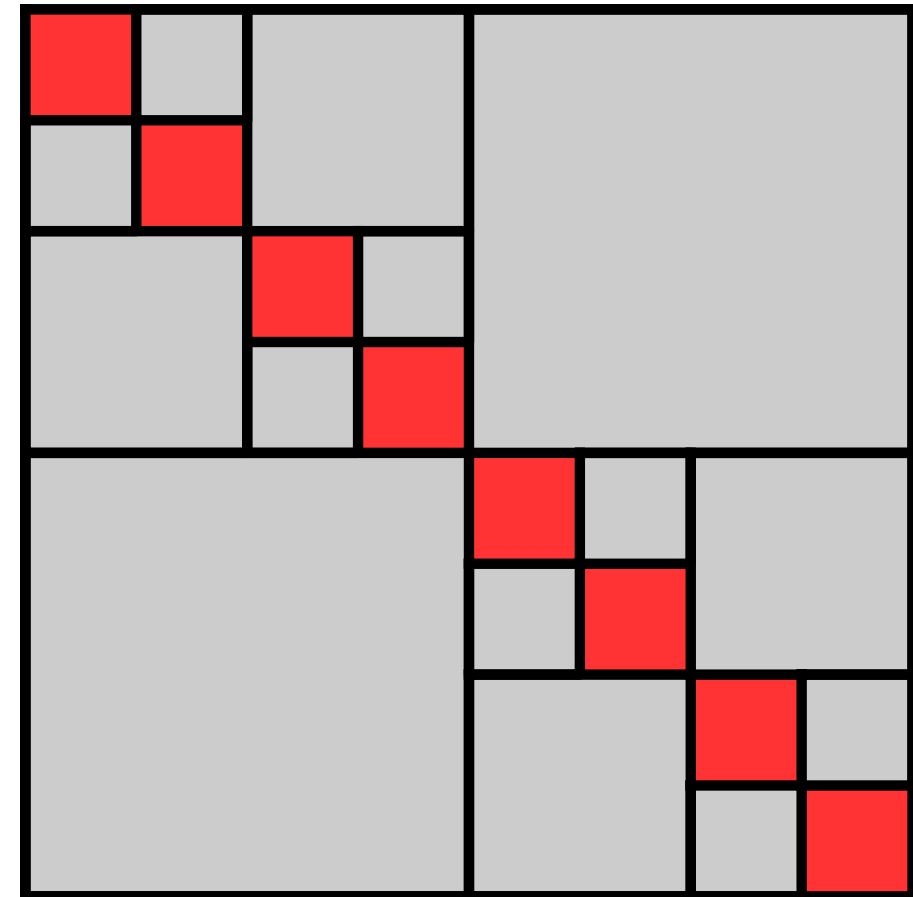
Obvious recursion!

Recall that we are interested in solving the PDE
$$\begin{cases} Au(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ Bu(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases} \quad (\text{BVP})$$

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

Recurring idea: Upon discretization, (SLN) leads to a matrix with *off-diagonal blocks of low numerical rank*.

This property can be exploited to attain linear or close to linear complexity for operations such as matrix-vector multiply, matrix-matrix multiply, LU factorization, matrix inversion, forming of Schur complements, etc.



All gray blocks have low rank.

Strong connections to Calderón-Zygmund theory for singular integral operators.

References: Fast Multipole Method (Greengard, Rokhlin); Panel Clustering (Hackbusch); \mathcal{H} - and \mathcal{H} -matrices; Hierarchically Block Separable (HBS) matrices; Hierarchically Semi Separable (HSS) matrices; S-matrices, a.k.a. HODLR matrices;...

In real life, tessellation patterns of the matrices that need to be inverted tend to be more complex ...

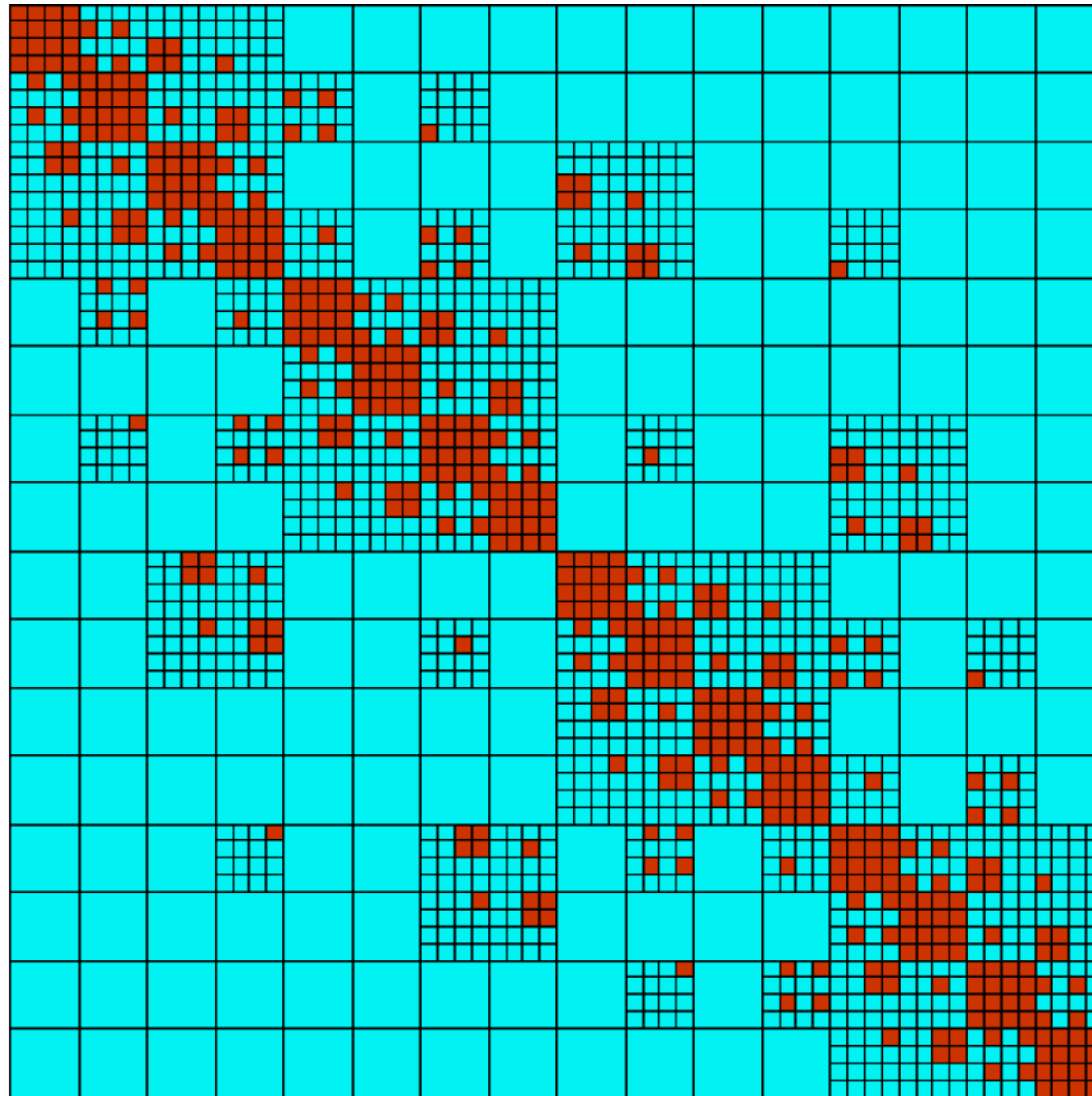


Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

Recall that we are interested in solving the PDE
$$\begin{cases} Au(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ Bu(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases} \quad (\text{BVP})$$

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

Question: Why do the dense matrices resulting upon discretization of (SLN) typically have *off-diagonal blocks of low numerical rank*?

(One) Answer: It 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.*

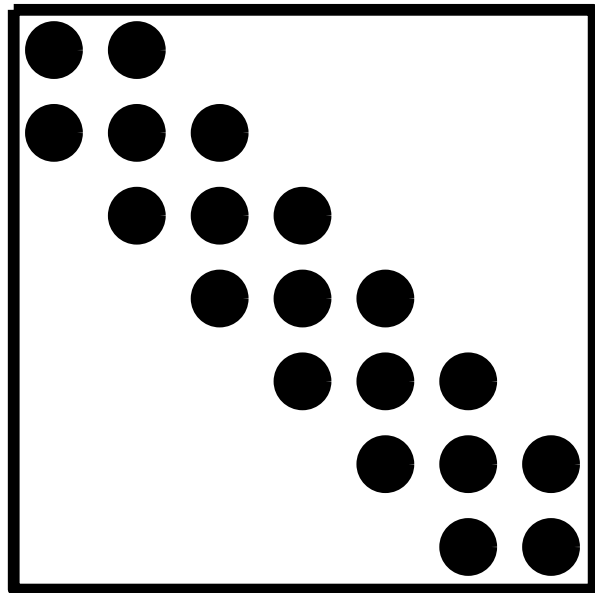
A 1D model problem: Consider a simple 2-point BVP on the interval $[0, 1]$:

$$(BVP) \quad \begin{cases} -\frac{d^2u}{dx^2} + p(x)\frac{du(x)}{dx} + q(x)u(x) = g(x), & x \in (0, 1), \\ u(0) = f_L, \\ u(1) = f_R. \end{cases}$$

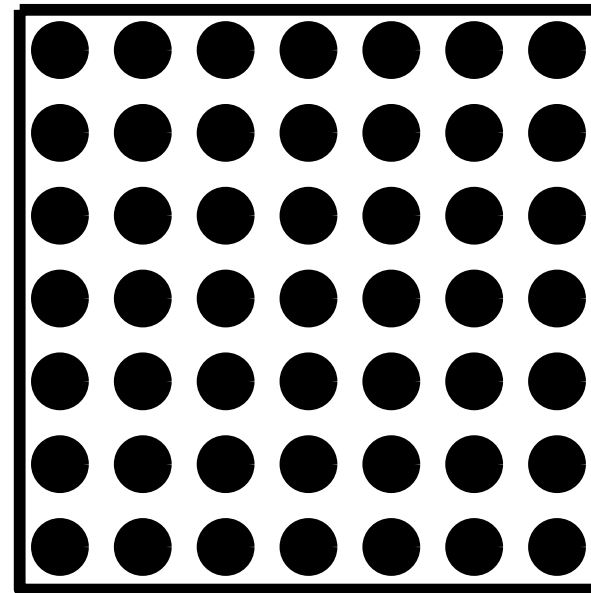
Discretizing (BVP) using a standard second order finite difference scheme, we get

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

where \mathbf{A} is a sparse matrix of size, say, $n \times n$. Then \mathbf{A}^{-1} is dense.



Sparsity pattern of \mathbf{A} .



Sparsity pattern of \mathbf{A}^{-1} .

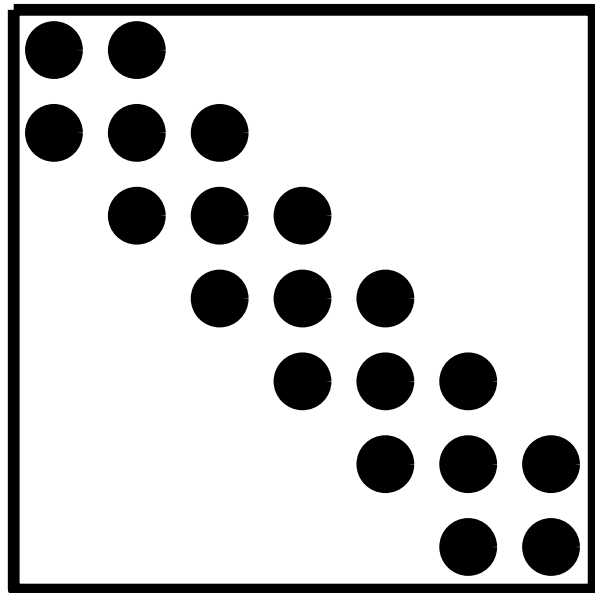
A 1D model problem: Consider a simple 2-point BVP on the interval $[0, 1]$:

$$(BVP) \quad \begin{cases} -\frac{d^2u}{dx^2} + p(x)\frac{du(x)}{dx} + q(x)u(x) = g(x), & x \in (0, 1), \\ u(0) = f_L, \\ u(1) = f_R. \end{cases}$$

Discretizing (BVP) using a standard second order finite difference scheme, we get

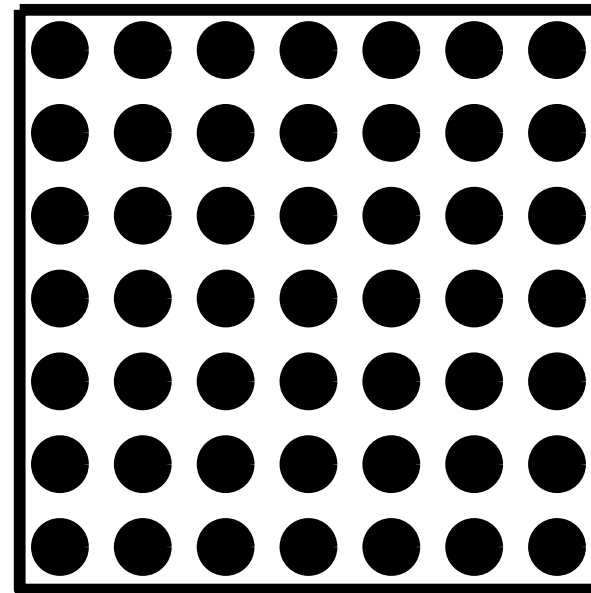
$$\mathbf{A}\mathbf{u} = \mathbf{b},$$

where \mathbf{A} is a sparse matrix of size, say, $n \times n$. Then \mathbf{A}^{-1} is dense.



Sparsity pattern of \mathbf{A} .

\mathbf{A} is tridiagonal.



Sparsity pattern of \mathbf{A}^{-1} .

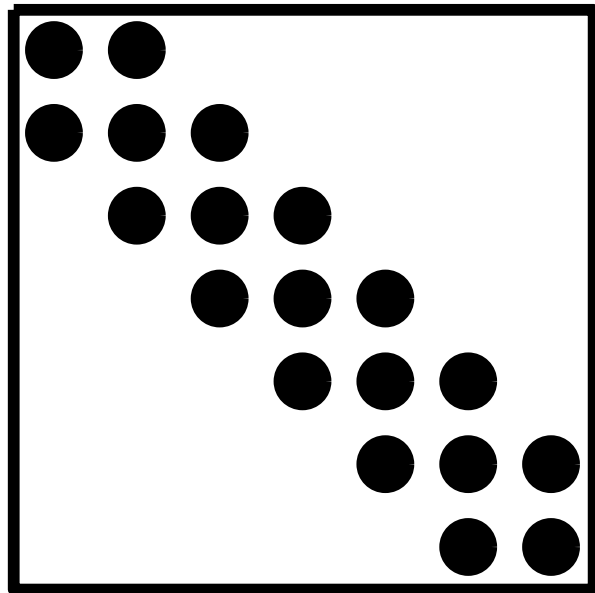
A 1D model problem: Consider a simple 2-point BVP on the interval $[0, 1]$:

$$(BVP) \quad \begin{cases} -\frac{d^2u}{dx^2} + p(x)\frac{du(x)}{dx} + q(x)u(x) = g(x), & x \in (0, 1), \\ u(0) = f_L, \\ u(1) = f_R. \end{cases}$$

Discretizing (BVP) using a standard second order finite difference scheme, we get

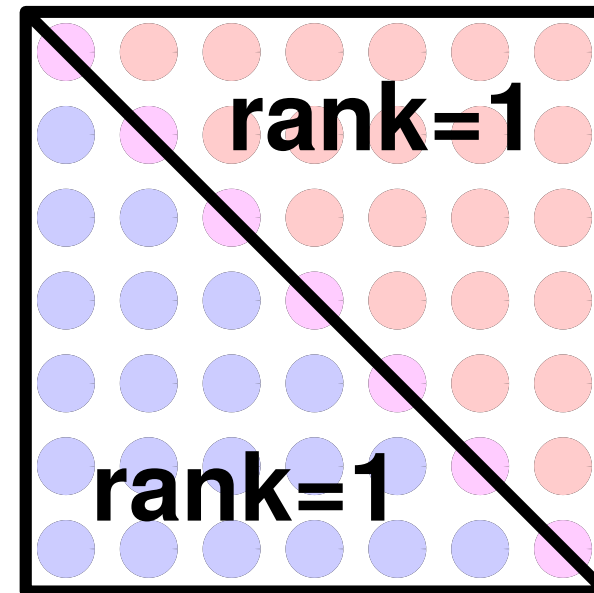
$$\mathbf{A}\mathbf{u} = \mathbf{b},$$

where \mathbf{A} is a sparse matrix of size, say, $n \times n$. Then \mathbf{A}^{-1} is dense.



Sparsity pattern of \mathbf{A} .

\mathbf{A} is tridiagonal.



Sparsity pattern of \mathbf{A}^{-1} .

\mathbf{A}^{-1} is semi-separable.

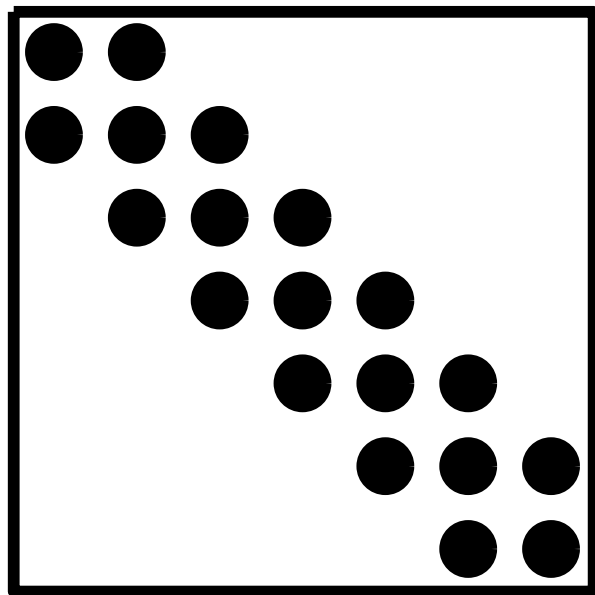
A 1D model problem: Consider a simple 2-point BVP on the interval $[0, 1]$:

$$(BVP) \quad \begin{cases} -\frac{d^2u}{dx^2} + p(x)\frac{du(x)}{dx} + q(x)u(x) = g(x), & x \in (0, 1), \\ u(0) = f_L, \\ u(1) = f_R. \end{cases}$$

Discretizing (BVP) using a standard second order finite difference scheme, we get

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

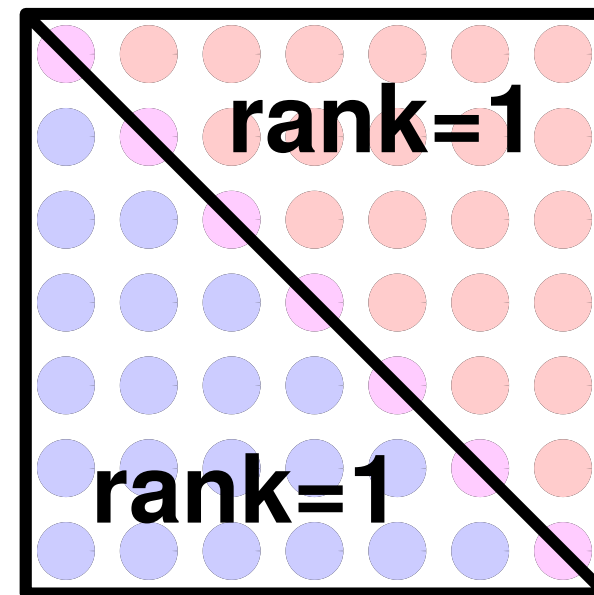
where \mathbf{A} is a sparse matrix of size, say, $n \times n$. Then \mathbf{A}^{-1} is dense.



Sparsity pattern of \mathbf{A} .

\mathbf{A} is tridiagonal.

\mathbf{A} is *sparse*.



Sparsity pattern of \mathbf{A}^{-1} .

\mathbf{A}^{-1} is semi-separable.

\mathbf{A}^{-1} is *data-sparse*.

Template:

- Consider an elliptic PDE

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

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

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

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

The matrix \mathbf{A} will be *sparse*.

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

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

The matrix \mathbf{S} will be *dense*, but *“data-sparse.”*

A 2D model problem: Let $\Omega = [0, 1]^2$ and $\Gamma = \partial\Omega$. We seek to solve

$$(8) \quad \begin{cases} -\Delta u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases}$$

We introduce an $n \times n$ grid on Ω with nodes $\{\mathbf{x}_j\}_{j=1}^N$ where $N = n^2$, see Figure A. Letting $\mathbf{u} = [\mathbf{u}(j)]_{j=1}^N$ denote a vector of approximate solution values, $\mathbf{u}(j) \approx u(\mathbf{x}_j)$, and using the standard five-point stencil to discretize $-\Delta$, we end up with a sparse linear system

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

where $[\mathbf{A}\mathbf{u}](k) = \frac{1}{h^2}(4\mathbf{u}(k) - \mathbf{u}(k_s) - \mathbf{u}(k_e) - \mathbf{u}(k_n) - \mathbf{u}(k_w))$, see Figure B.

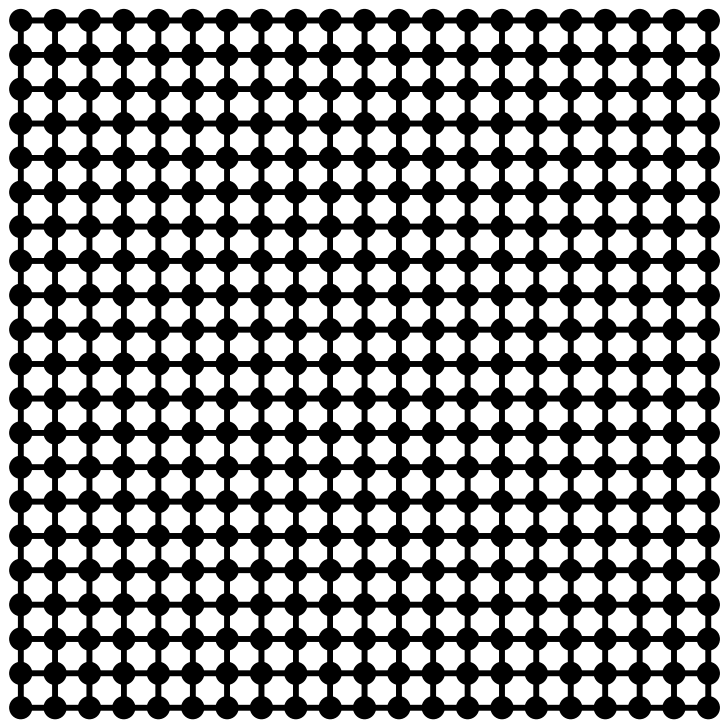


Figure A: The grid

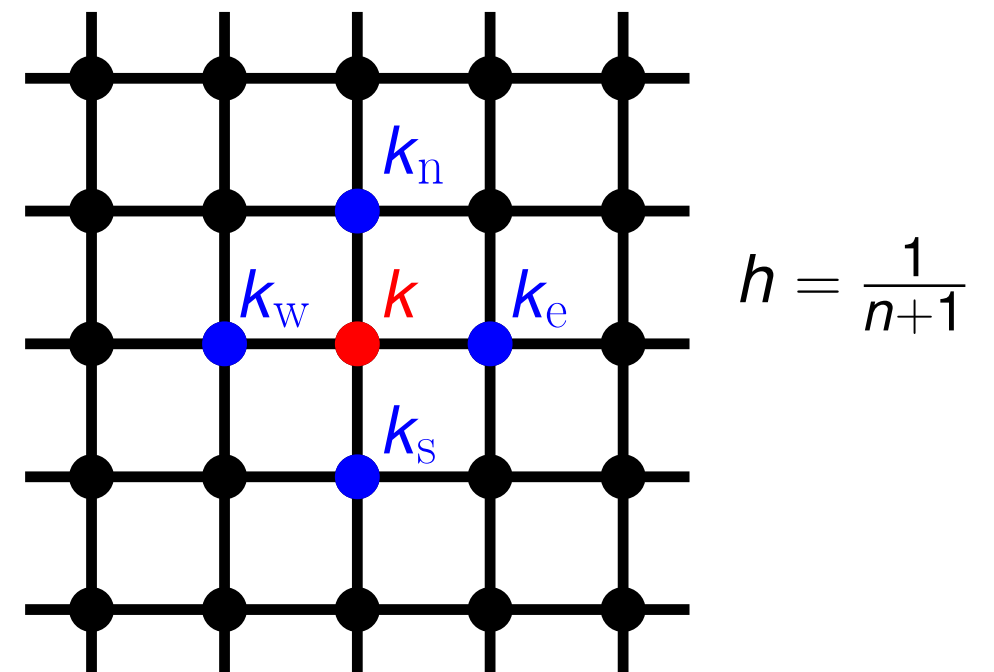
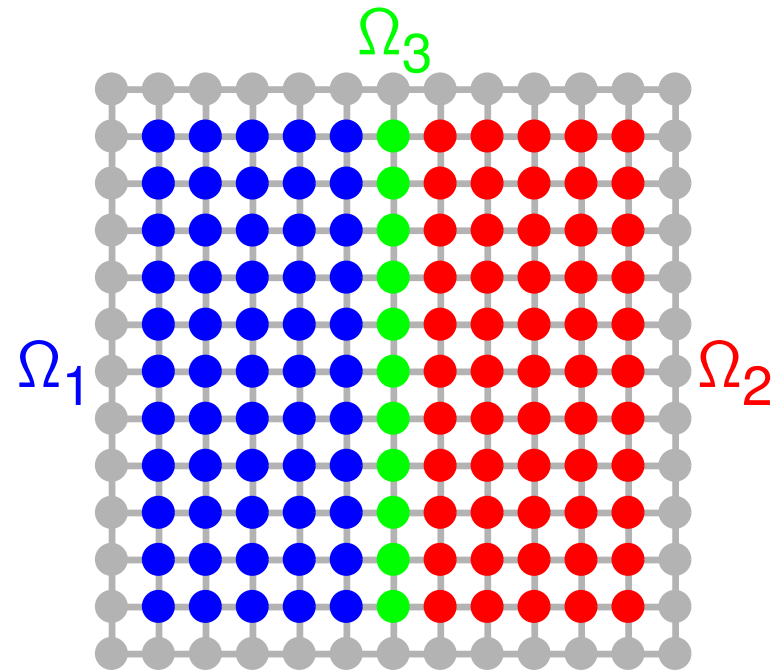


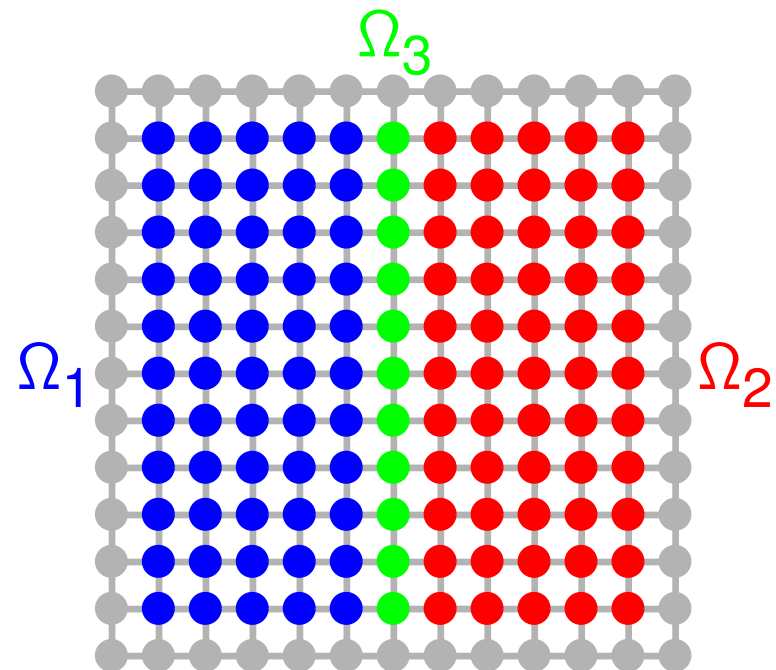
Figure B: The 5-point stencil

Divide-and-conquer: Split the nodes in three groups as shown so that there are no connections between nodes in Ω_1 and Ω_2 . Then \mathbf{A} has zero blocks as shown:



$$\mathbf{A} = \begin{array}{|c|c|c|} \hline \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \hline \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \hline \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \\ \hline \end{array}$$

Divide-and-conquer: Split the nodes in three groups as shown so that there are no connections between nodes in Ω_1 and Ω_2 . Then \mathbf{A} has zero blocks as shown:



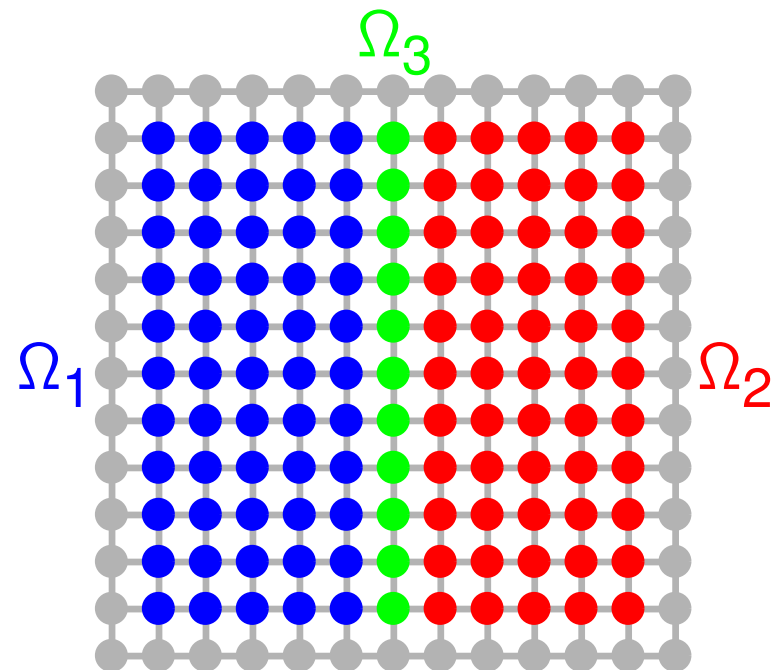
$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$$

Now suppose that we can somehow construct \mathbf{A}_{11}^{-1} and \mathbf{A}_{22}^{-1} . Then

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{31}\mathbf{A}_{11}^{-1} & \mathbf{A}_{32}\mathbf{A}_{22}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{A}_{13} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

where $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$ is a *Schur complement*.

Divide-and-conquer: Split the nodes in three groups as shown so that there are no connections between nodes in Ω_1 and Ω_2 . Then \mathbf{A} has zero blocks as shown:



$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$$

Now suppose that we can somehow construct \mathbf{A}_{11}^{-1} and \mathbf{A}_{22}^{-1} . Then

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{31}\mathbf{A}_{11}^{-1} & \mathbf{A}_{32}\mathbf{A}_{22}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{A}_{13} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

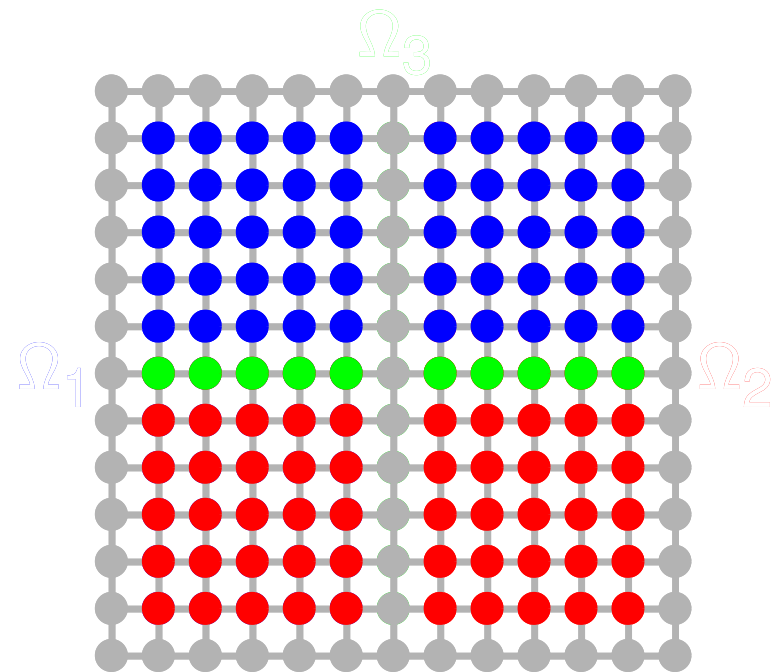
where $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$ is a *Schur complement*.

In other words, in order to invert \mathbf{A} , we need to execute three steps:

- Invert \mathbf{A}_{11} to form \mathbf{A}_{11}^{-1} . *size* $\sim N/2 \times N/2$
- Invert \mathbf{A}_{22} to form \mathbf{A}_{22}^{-1} . *size* $\sim N/2 \times N/2$
- Invert $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$. *size* $\sim \sqrt{N} \times \sqrt{N}$

Notice the obvious recursion!

Divide-and-conquer: Split the nodes in three groups as shown so that there are no connections between nodes in Ω_1 and Ω_2 . Then \mathbf{A} has zero blocks as shown:



$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$$

Now suppose that we can somehow construct \mathbf{A}_{11}^{-1} and \mathbf{A}_{22}^{-1} . Then

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{31}\mathbf{A}_{11}^{-1} & \mathbf{A}_{32}\mathbf{A}_{22}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{A}_{13} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

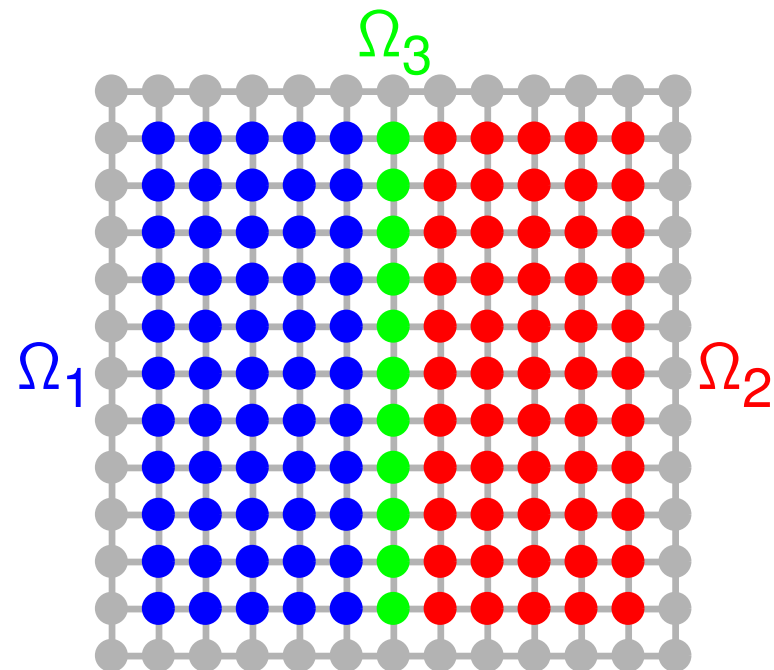
where $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$ is a *Schur complement*.

In other words, in order to invert \mathbf{A} , we need to execute three steps:

- Invert \mathbf{A}_{11} to form \mathbf{A}_{11}^{-1} . *size* $\sim N/2 \times N/2$
- Invert \mathbf{A}_{22} to form \mathbf{A}_{22}^{-1} . *size* $\sim N/2 \times N/2$
- Invert $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$. *size* $\sim \sqrt{N} \times \sqrt{N}$

Notice the obvious recursion!

Divide-and-conquer: Split the nodes in three groups as shown so that there are no connections between nodes in Ω_1 and Ω_2 . Then \mathbf{A} has zero blocks as shown:



$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$$

Now suppose that we can somehow factor $\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{U}_{11}$ and $\mathbf{A}_{22} = \mathbf{L}_{22}\mathbf{U}_{22}$. Then

$$\mathbf{A} = \begin{bmatrix} \mathbf{L}_{11}\mathbf{U}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{L}_{22}\mathbf{U}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{11} & \mathbf{0} & \\ \mathbf{0} & \mathbf{L}_{22} & \\ \mathbf{A}_{31}\mathbf{U}_{11}^{-1} & \mathbf{A}_{32}\mathbf{U}_{22}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{11} & \mathbf{0} & \mathbf{L}_{11}^{-1}\mathbf{A}_{13} \\ \mathbf{0} & \mathbf{U}_{22} & \mathbf{L}_{22}^{-1}\mathbf{A}_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

where $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{U}_{22}^{-1}\mathbf{L}_{22}^{-1}\mathbf{A}_{23}$ is a *Schur complement*.

In other words, in order to invert \mathbf{A} , we need to execute three steps:

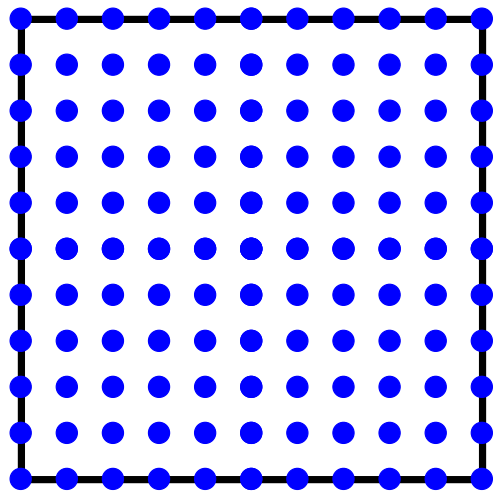
- Factor \mathbf{A}_{11} to form $\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{U}_{11}$. *size* $\sim N/2 \times N/2$
- Factor \mathbf{A}_{22} to form $\mathbf{A}_{22} = \mathbf{L}_{22}\mathbf{U}_{22}$. *size* $\sim N/2 \times N/2$
- Factor $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{U}_{22}^{-1}\mathbf{L}_{22}^{-1}\mathbf{A}_{23}$. *size* $\sim \sqrt{N} \times \sqrt{N}$

Notice the obvious recursion!

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.



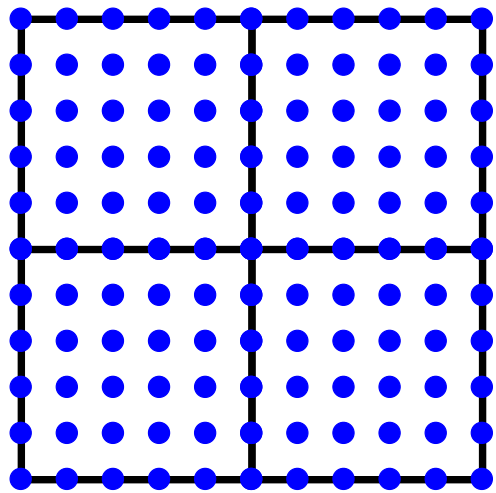
The original grid.

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.

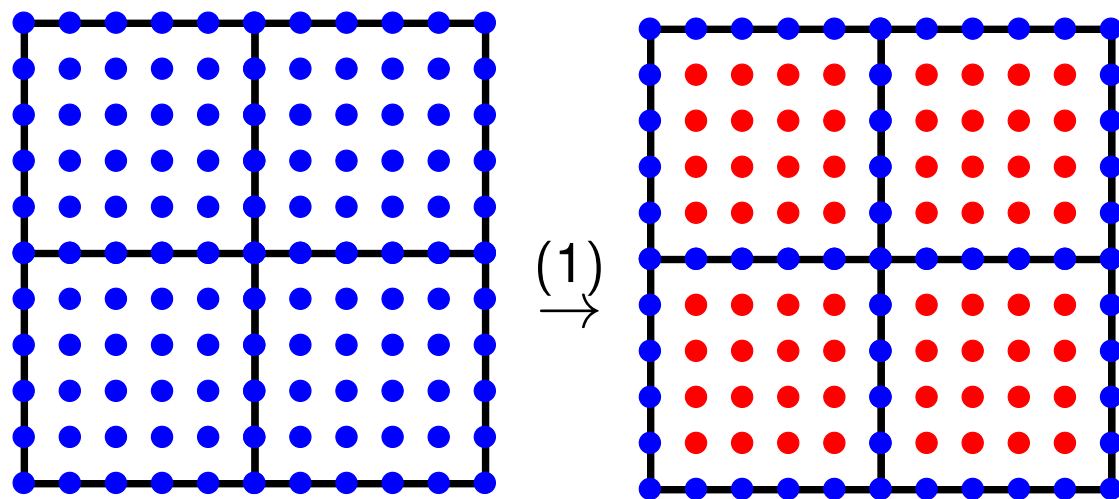
Split the domain into “small” patches we call “leaves” (they will be organized in a tree).



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

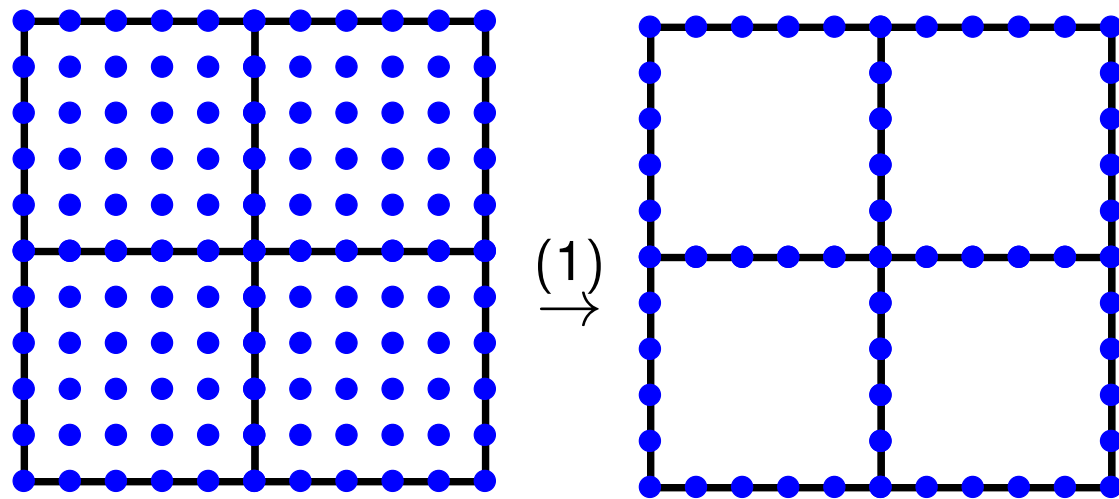
All direct solvers to be described are based on hierarchical domain decomposition.

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.

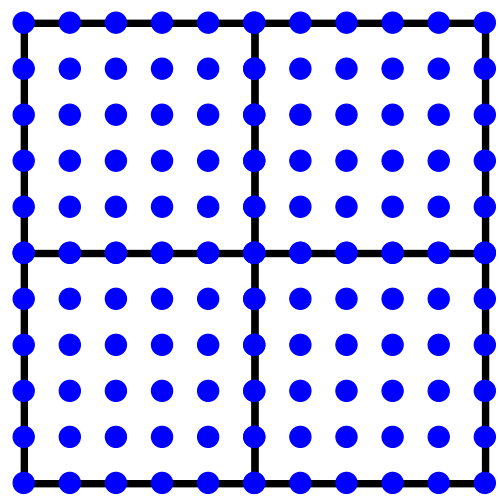
Consider a PDE $Au = f$ defined on a square $\Omega = [0, 1]$. Put a grid on the square.

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).

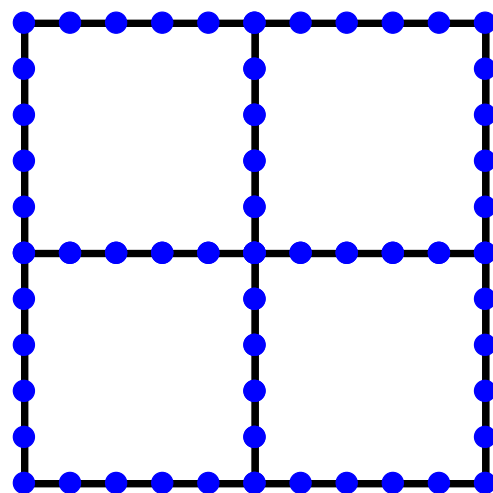
This eliminates “internal” grid points from the computation. (“Static condensation.”)

Merge the leaves in pairs of two.



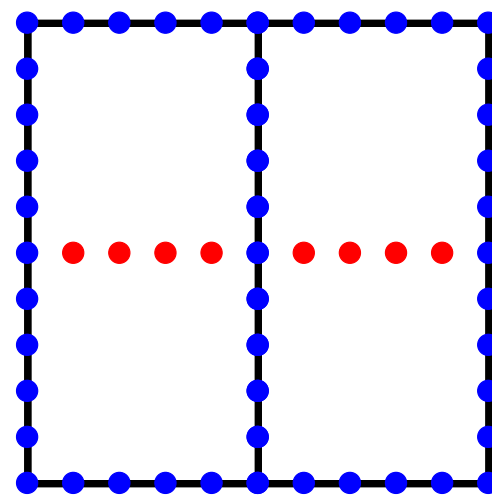
The original grid.

(1)
→



Leaves reduced.

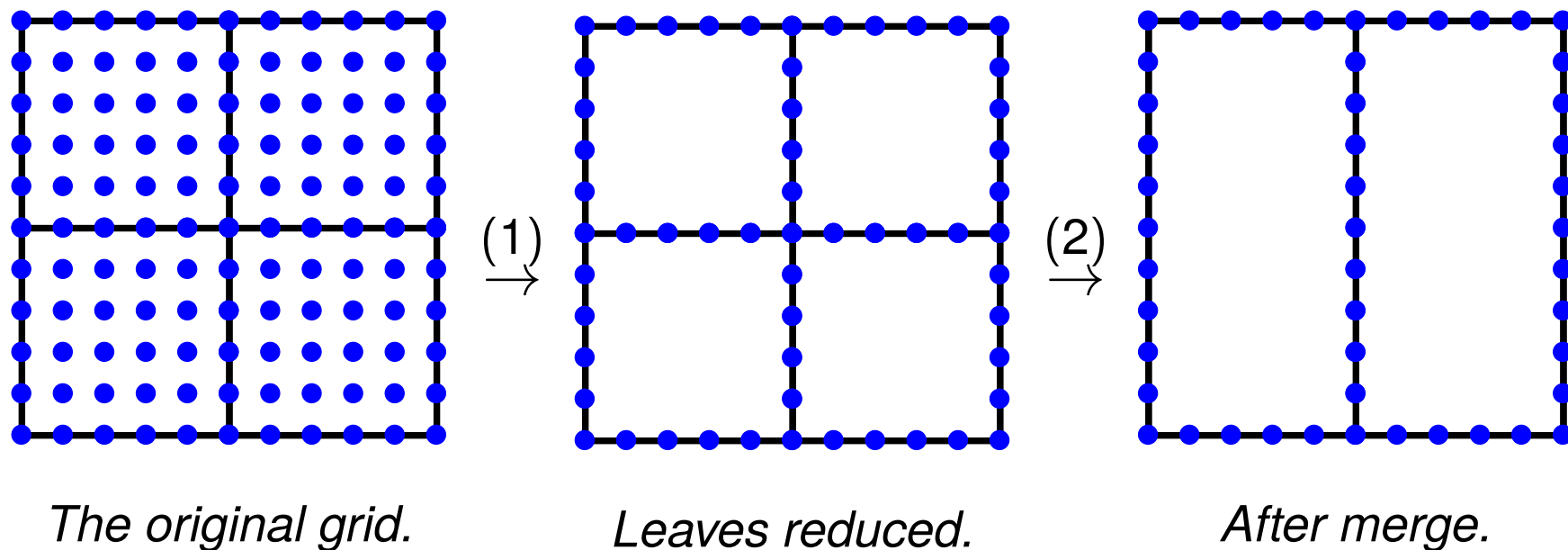
(2)
→



After merge.

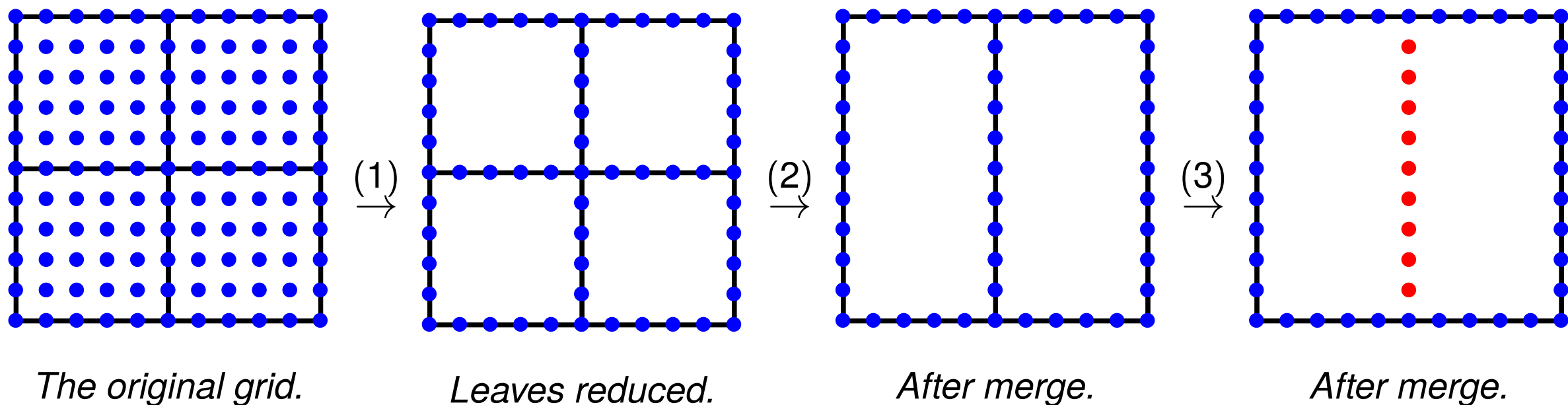
Outline of direct solver

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.



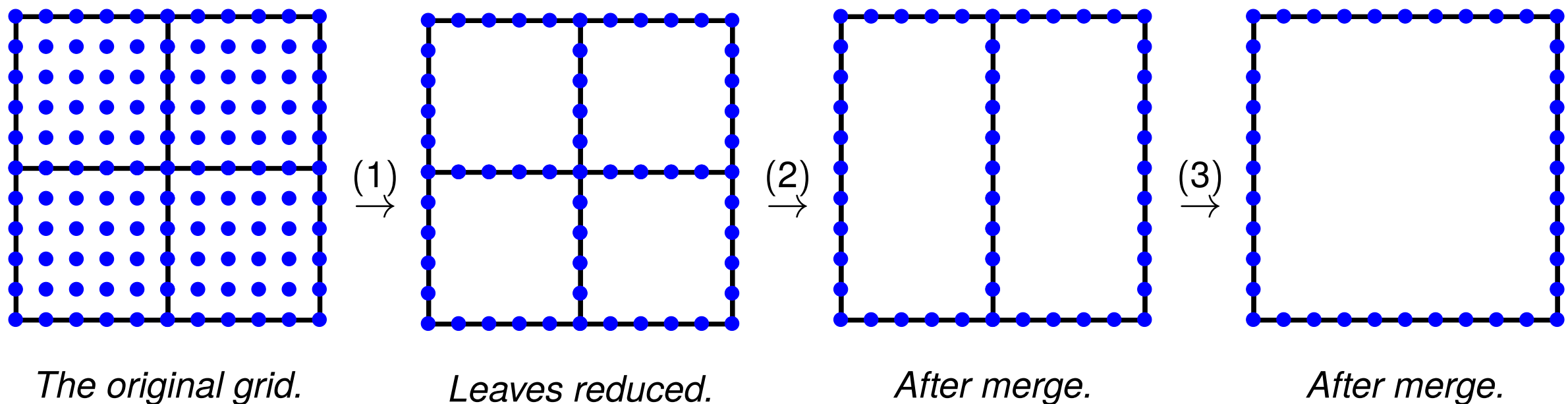
Outline of direct solver

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.



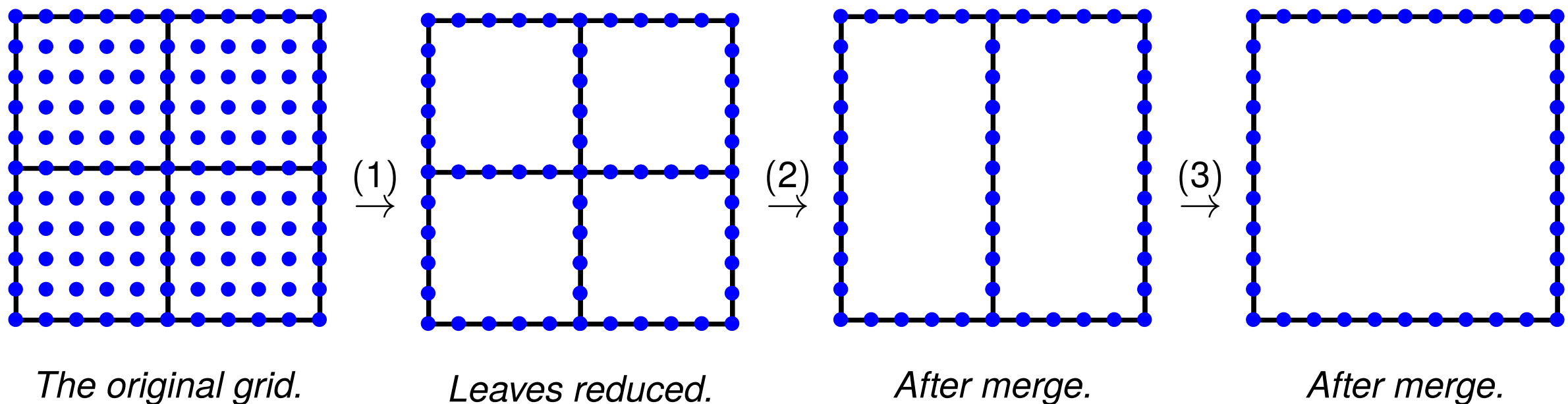
Outline of direct solver

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.



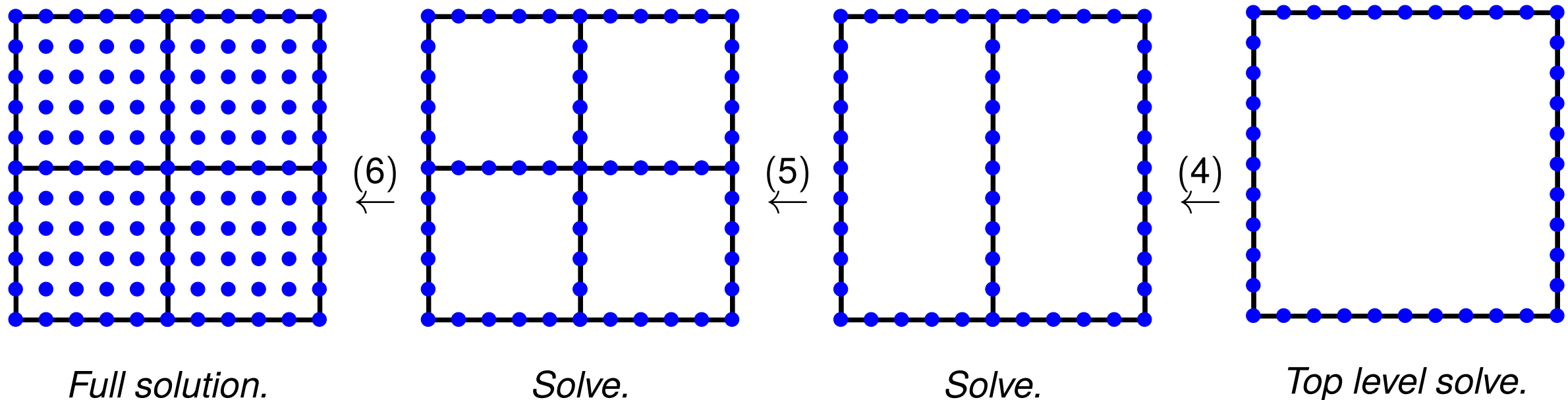
Outline of direct solver

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.

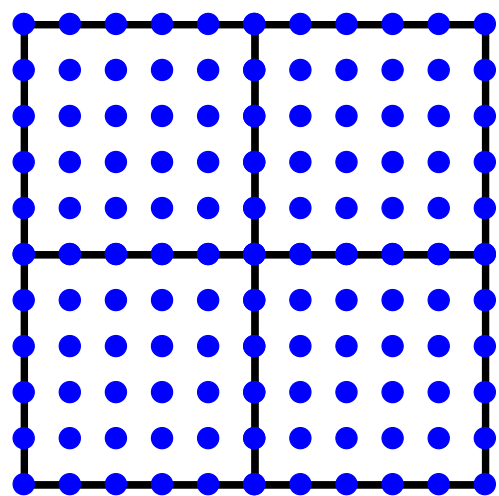


Outline of direct solver

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.

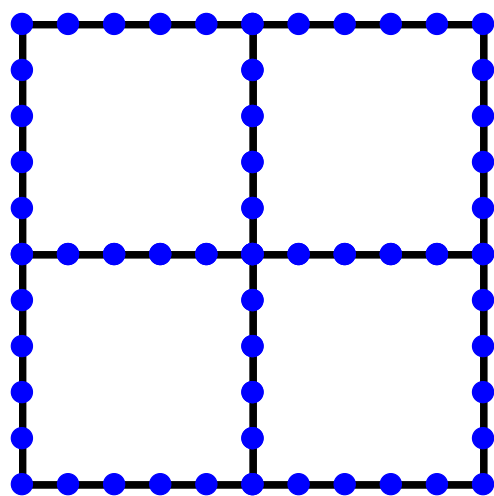


Upwards pass — build all solution operators:



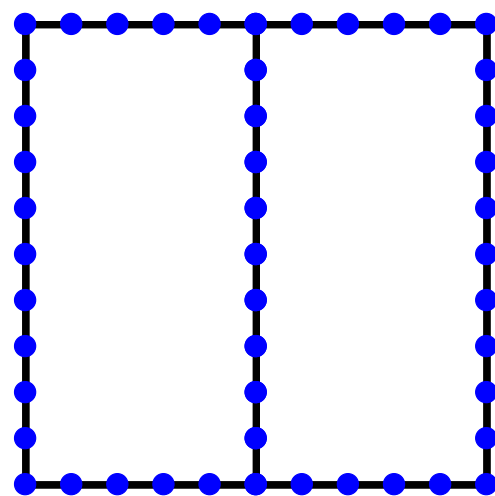
The original grid.

(1)
→



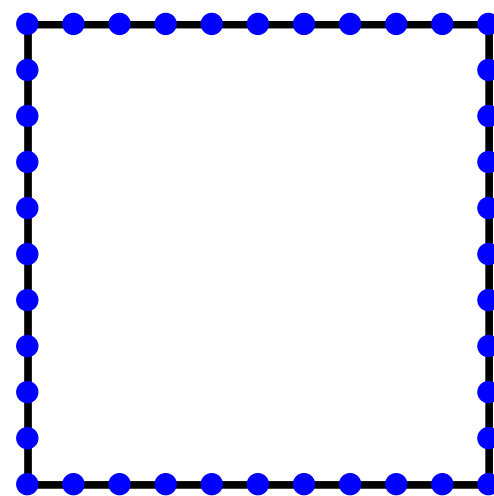
Leaves reduced.

(2)
→



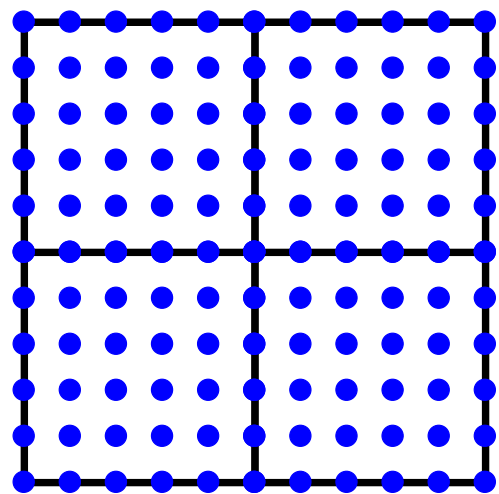
After merge.

(3)
→



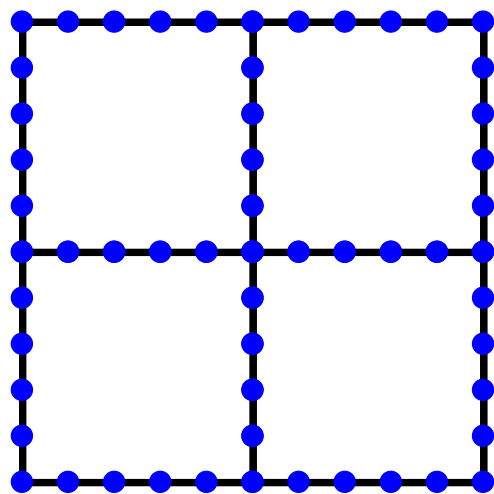
After merge.

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



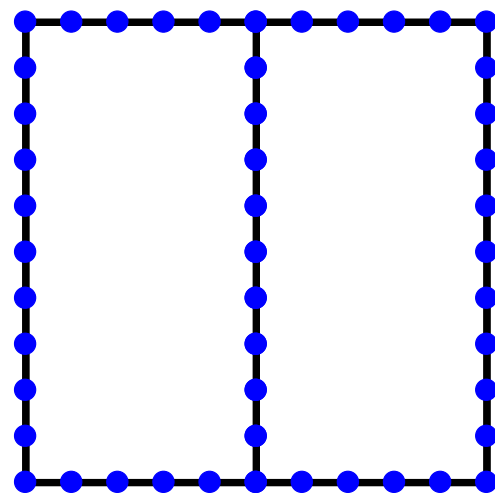
Full solution.

(6)
←



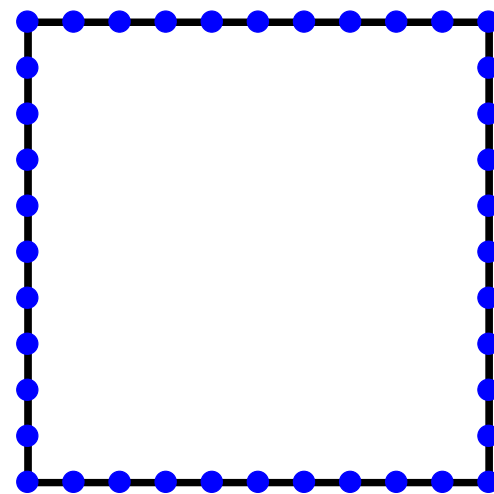
Solve.

(5)
←



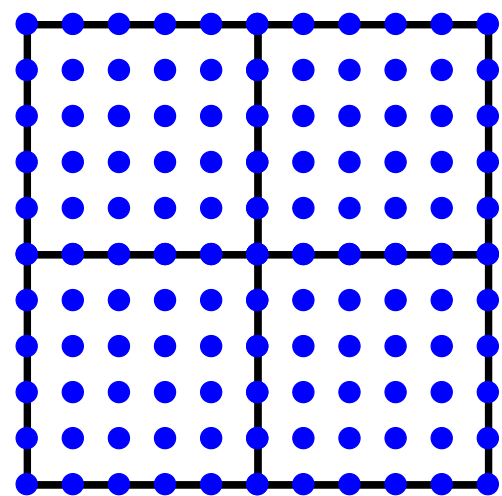
Solve.

(4)
←



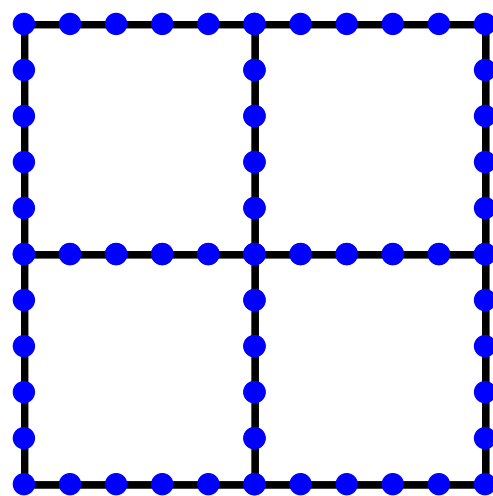
Top level solve.

Upwards pass — build all solution operators:



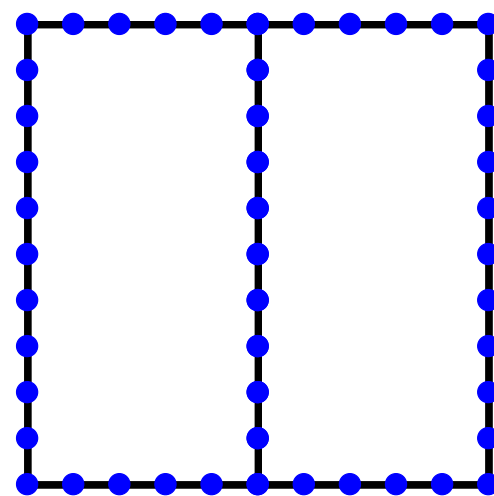
The original grid.

(1)
→



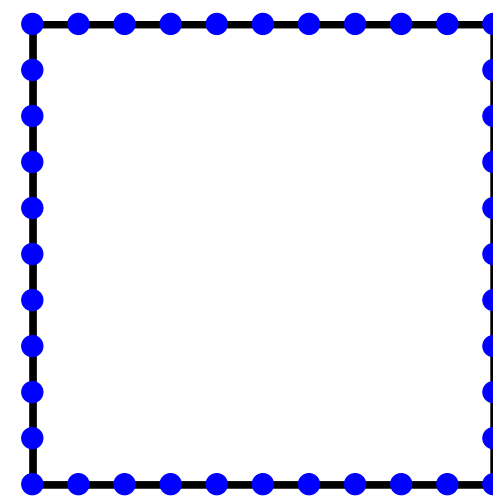
Leaves reduced.

(2)
→



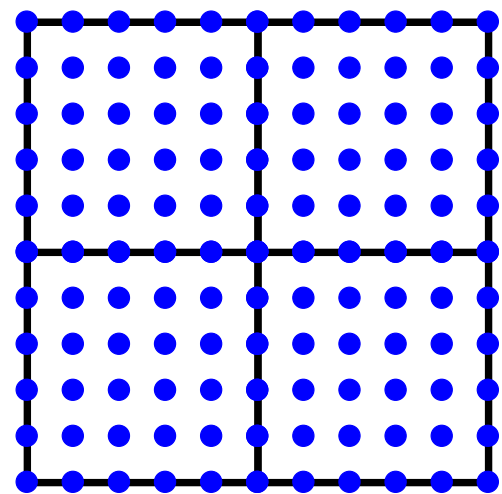
After merge.

(3)
→



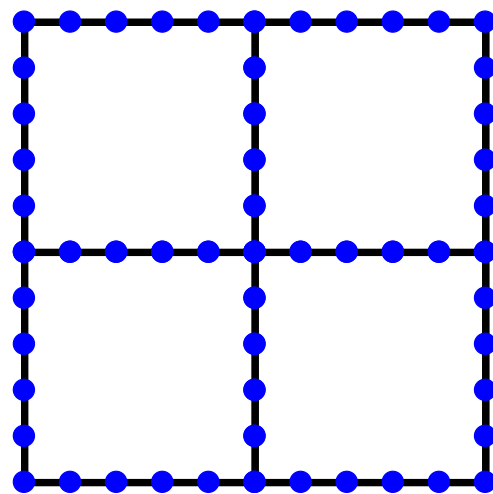
After merge.

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



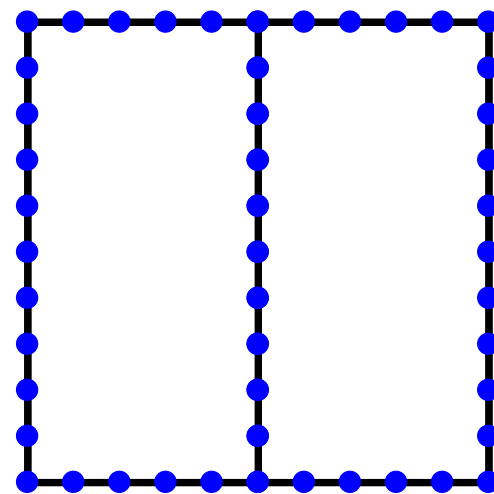
Full solution.

(6)
←



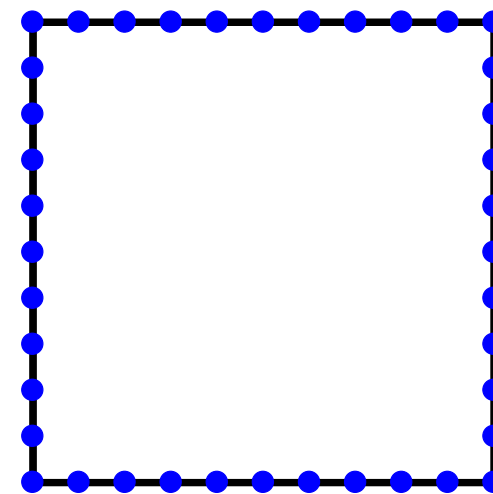
Solve.

(5)
←



Solve.

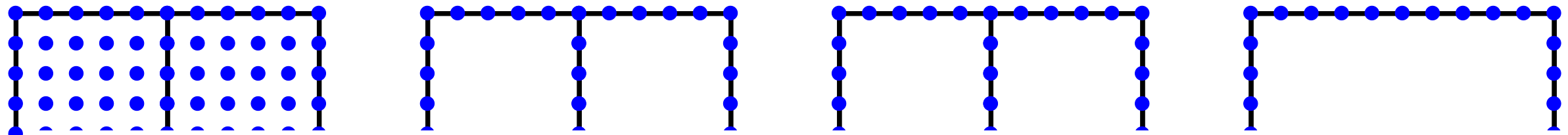
(4)
←



Top level solve.

Well-established idea: Classical multifrontal / nested dissection method (1973).

Upwards pass — build all solution operators:



D



Alan George



Iain Duff



Tim Davis

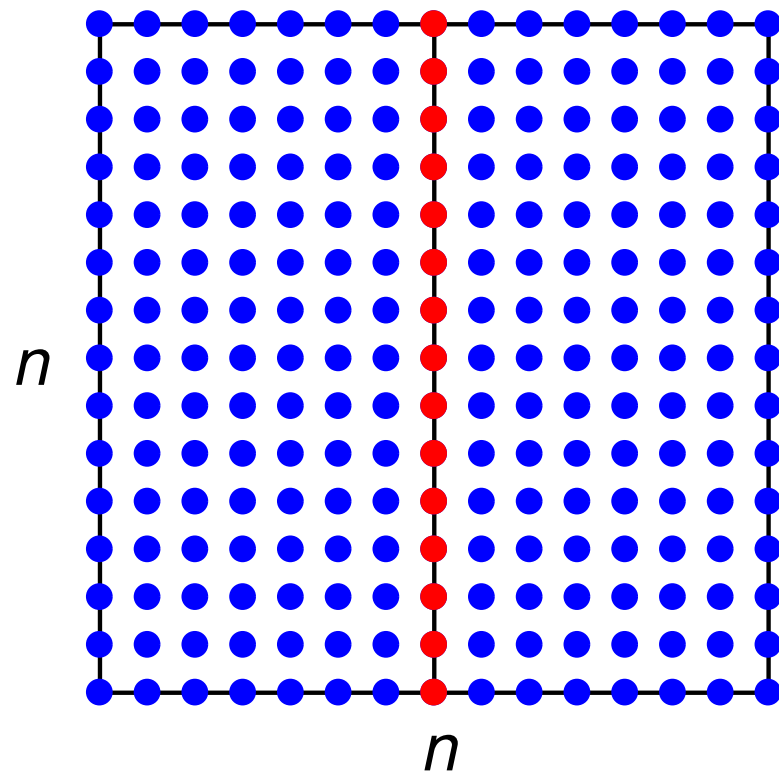
Well-established idea: Classical multifrontal / nested dissection method (1973).

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

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:



$$N = n \times n$$

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

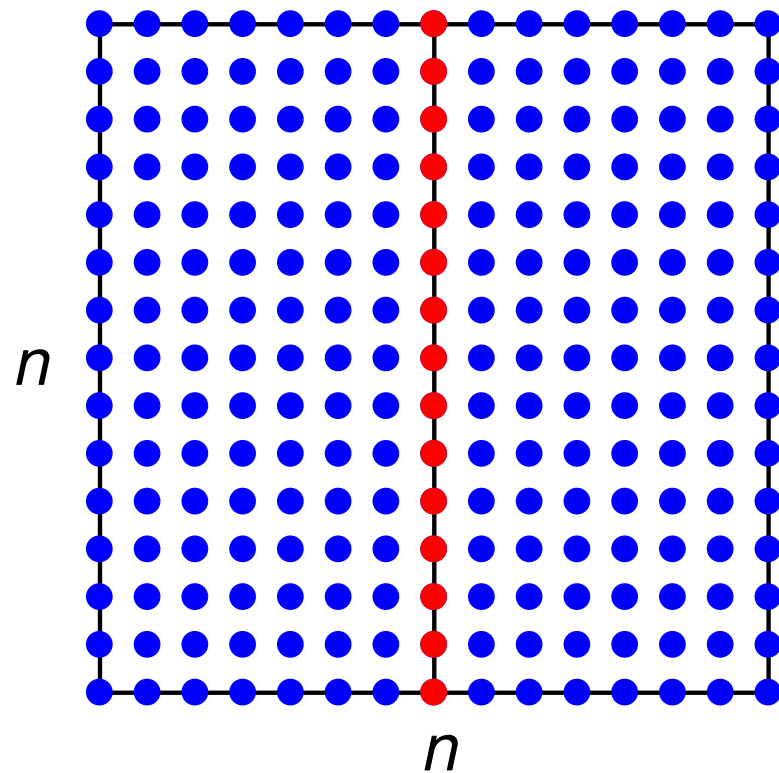
Since this dense matrix is of size $n \times n$, the cost for the merge is

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

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

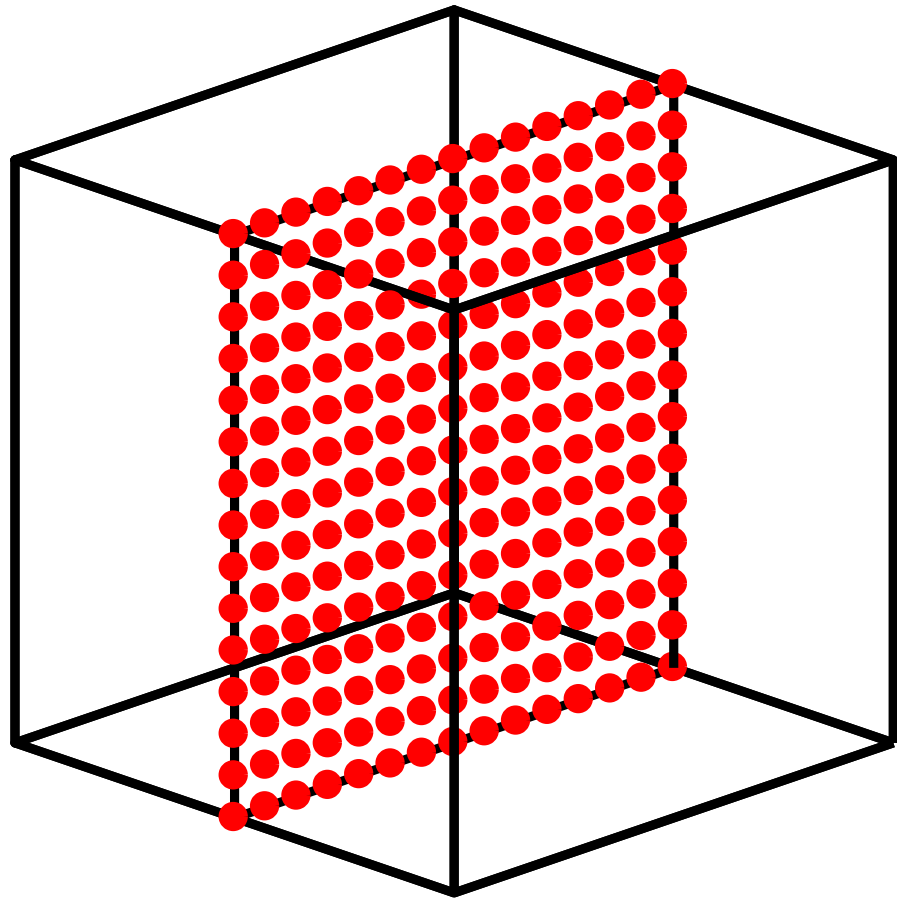
$$\text{COST} \sim n^3 \sim (N^{1/2})^3 \sim N^{3/2}.$$

Problem: 3D is much 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:



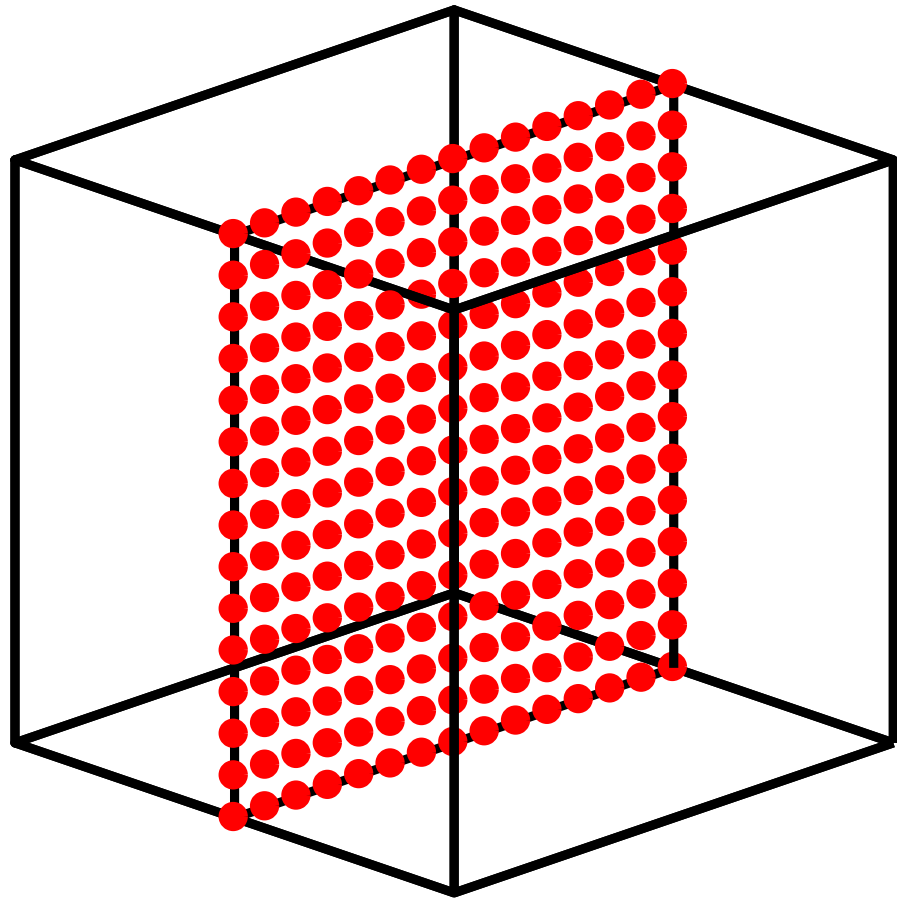
The merge requires factorization of a dense matrix of size $n^2 \times n^2$. Consequently:

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

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:



The merge requires factorization of a dense matrix of size $n^2 \times n^2$. Consequently:

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

Assertion: The dense matrix very often behaves like a discretized integral operator. (E.g. Dirichlet-to-Neumann.)

It is rank-structured, and is amenable to “fast” matrix algebra.

We can reduce the complexity of the top level solve from $O(N^2)$ down to $O(N)$, and sometimes even $O(N^{2/3})$.

Through exploiting the assertion on the previous page, the complexity of direct solvers for elliptic PDEs has in the past 10 – 20 years been decreased dramatically:

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

Through exploiting the assertion on the previous page, the complexity of direct solvers for elliptic PDEs has in the past 10 – 20 years been decreased dramatically:

	<i>Build stage</i>		<i>Solve stage</i>	
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), **J. Xia**, Chandrasekaran, Gu, & Li (2009), Gillman & Martinsson (2011), Schmitz & **L. Ying** (2012), Darve & Ambikasaran (2013), Ho & Ying (2015), Amestoy, Ashcraft, et al (2015), Oseledets & Suchnikova (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). Work in 1990's Y. Chen, P. Starr, **V. Rokhlin**, **L. Greengard**, **E. Michielssen**. Related to work on \mathcal{H} and \mathcal{H}^2 matrix methods (1998 and forwards) by Börm, Bebendorf, Hackbusch, Khoromskij, Sauter, 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.

Key selling point: Better parallelism

Let us consider the flop counts of various parts of the computation:

	Classical Nested Dissection	Accelerated Nested Dissection
Cost to process leaves:	$\sim N$	$\sim N$
Cost to process the root:	$\sim N^2$	$\sim N^{2/3}$

Observations:

- While the dominant cost of the old scheme is processing dense matrices of size $O(N^{2/3}) \times O(N^{2/3})$, the dominant cost of the new scheme is processing the leaves.

Key selling point: Better parallelism

Let us consider the flop counts of various parts of the computation:

	Classical Nested Dissection	Accelerated Nested Dissection
Cost to process leaves:	$\sim N$	$\sim N$
Cost to process the root:	$\sim N^2$	$\sim N^{2/3}$

Observations:

- While the dominant cost of the old scheme is processing dense matrices of size $O(N^{2/3}) \times O(N^{2/3})$, the dominant cost of the new scheme is processing the leaves.
- *The leaf computations are very easy to parallelize!*
- Parallel implementations of structured matrix algebra requires hard work (J. Poulson's dissertation; S. Li at LBNL; G. Biros; R. Kriemann; P. Amestoy, A. Buttari & T. Mary; G. Turkiyyah & D. Keyes; J. Xia; etc).
- For intermediate size problems, the structured matrices of size $O(N^{2/3}) \times O(N^{2/3})$ often fit on one machine.
- The methodology need not be all-or-nothing. Direct solvers can be used locally to handle areas with mesh refinement etc.

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

- Direct solvers use a lot of memory per degree of freedom.
→ *You want to maximize the oomph per DOF.*
- Direct solvers are particularly well suited for “high” frequency wave problems.
→ *Need high accuracy due to ill-conditioned physics.*
- High order methods sometimes lead to more ill-conditioned systems.
→ *Can be hard to get iterative solvers to converge.*

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

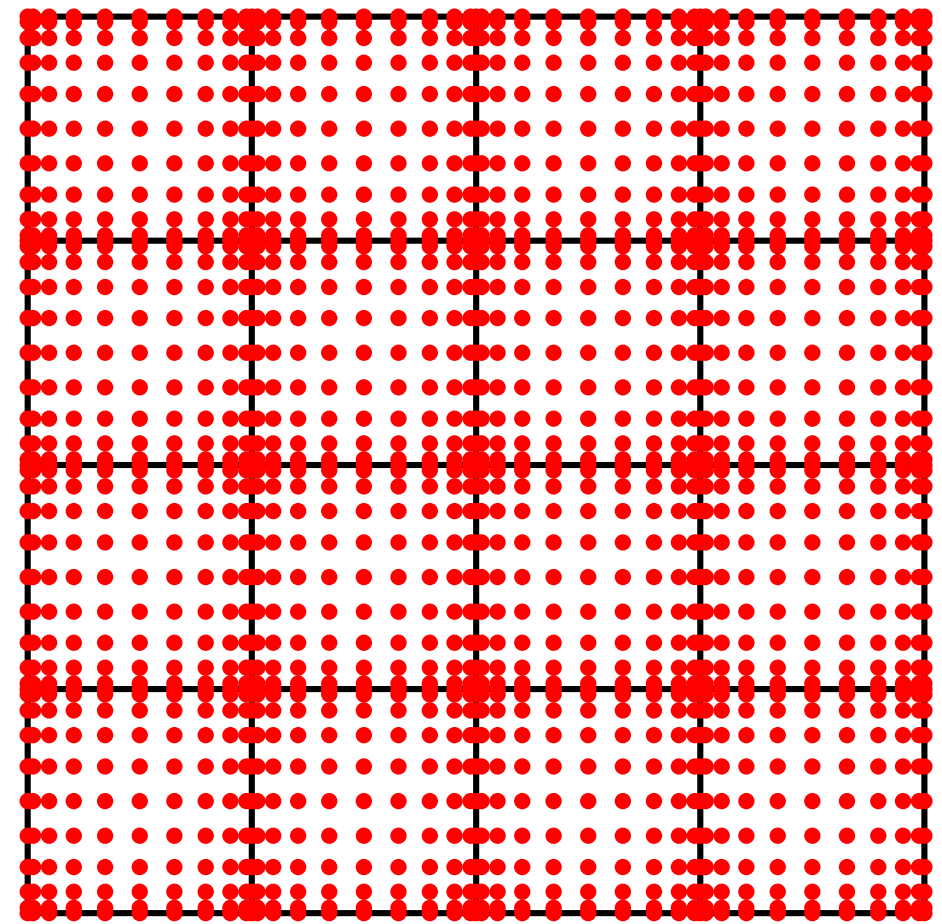
$$\begin{cases} -\Delta u(\mathbf{x}) - b(\mathbf{x}) u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases}$$

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).
Connections to domain decomposition and “reduction to interface” methods (Khoromskij & Wittum, etc.).

The Hierarchical Poincaré-Steklov Method

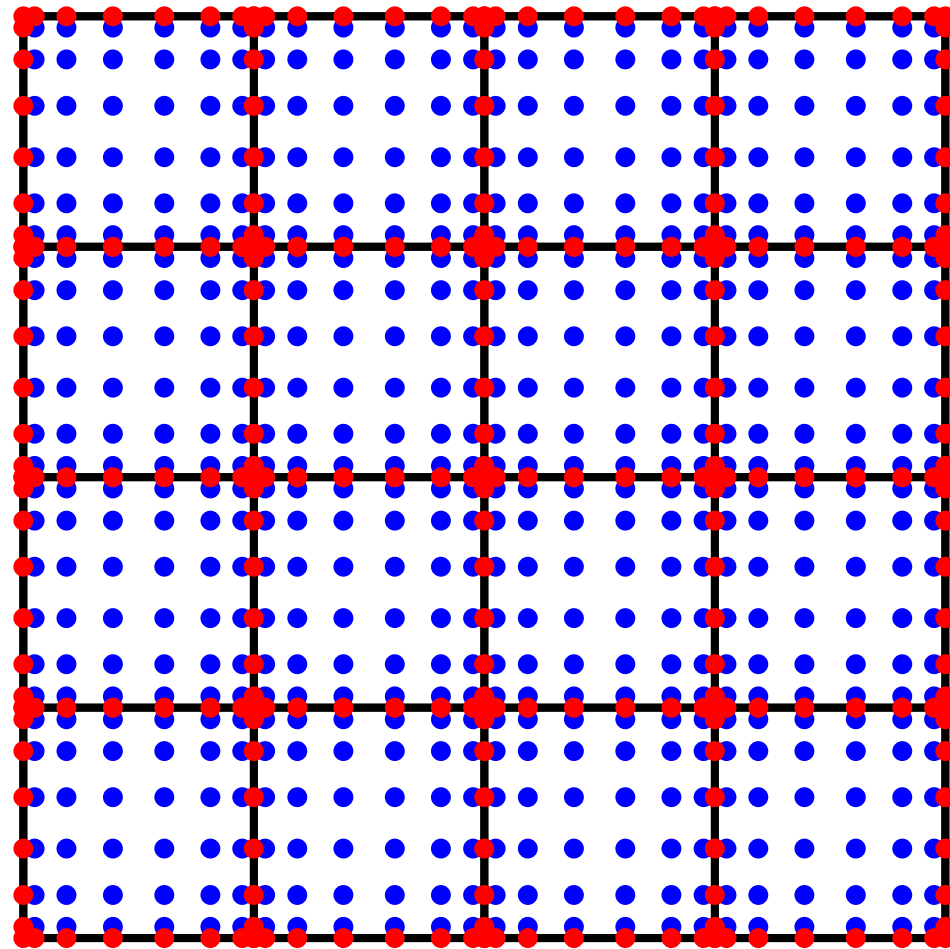
Model problem: Given f , g , and b , find u such that

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

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

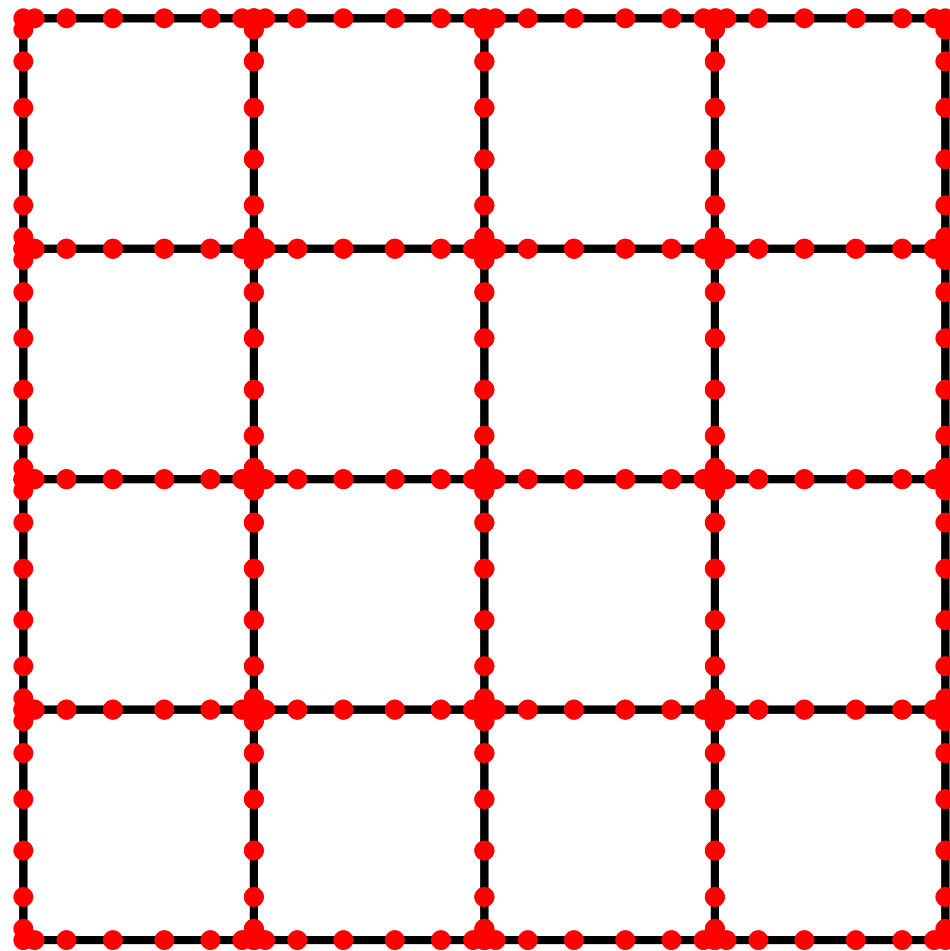
Model problem: Given f , g , and b , find u such that

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

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

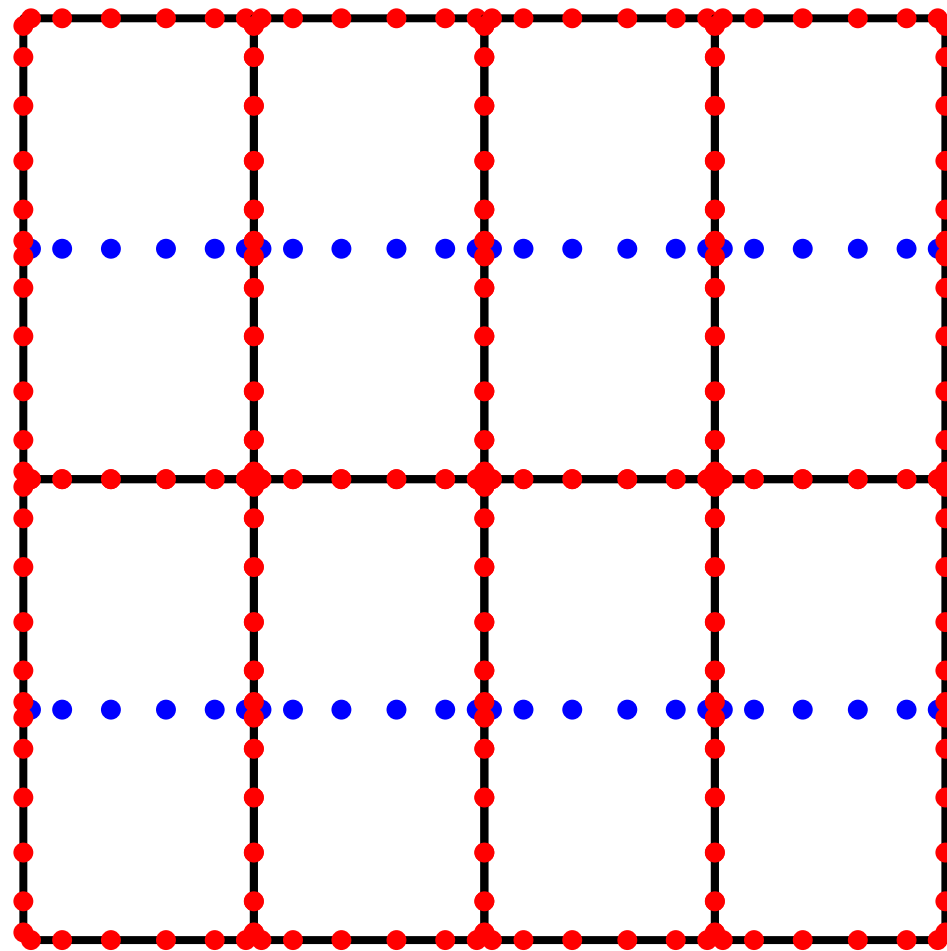
Model problem: Given f , g , and b , find u such that

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

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

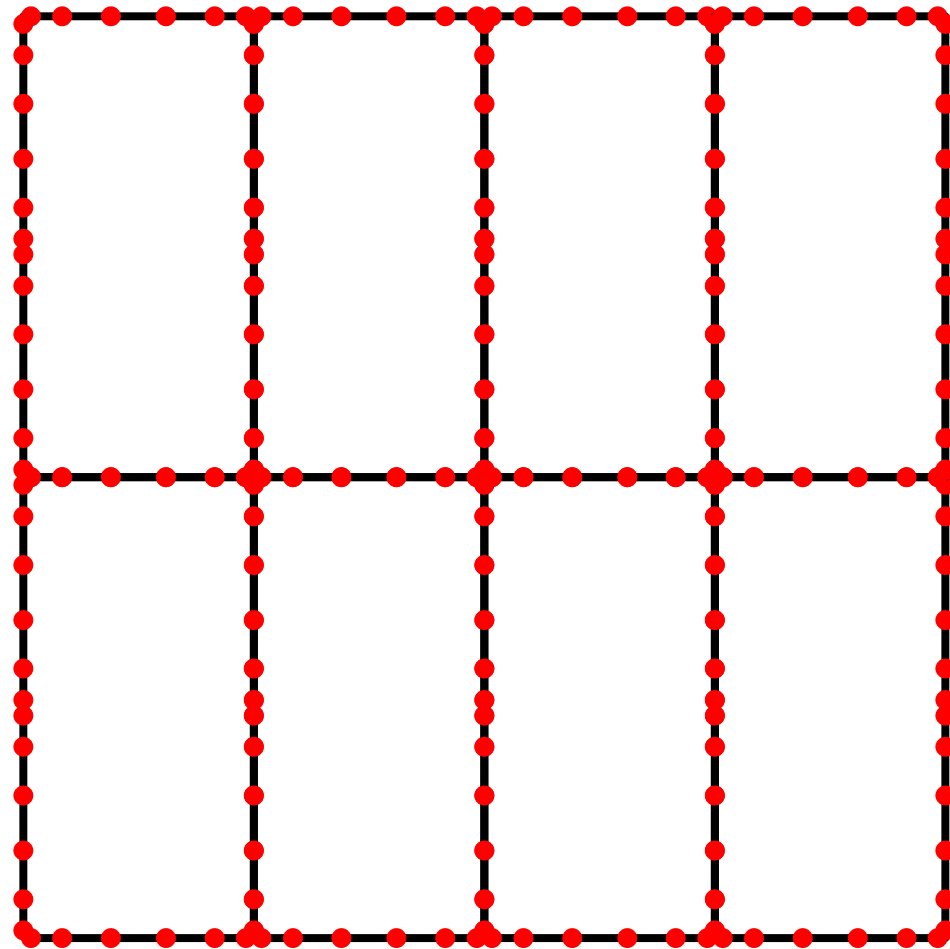
Model problem: Given f , g , and b , find u such that

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

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

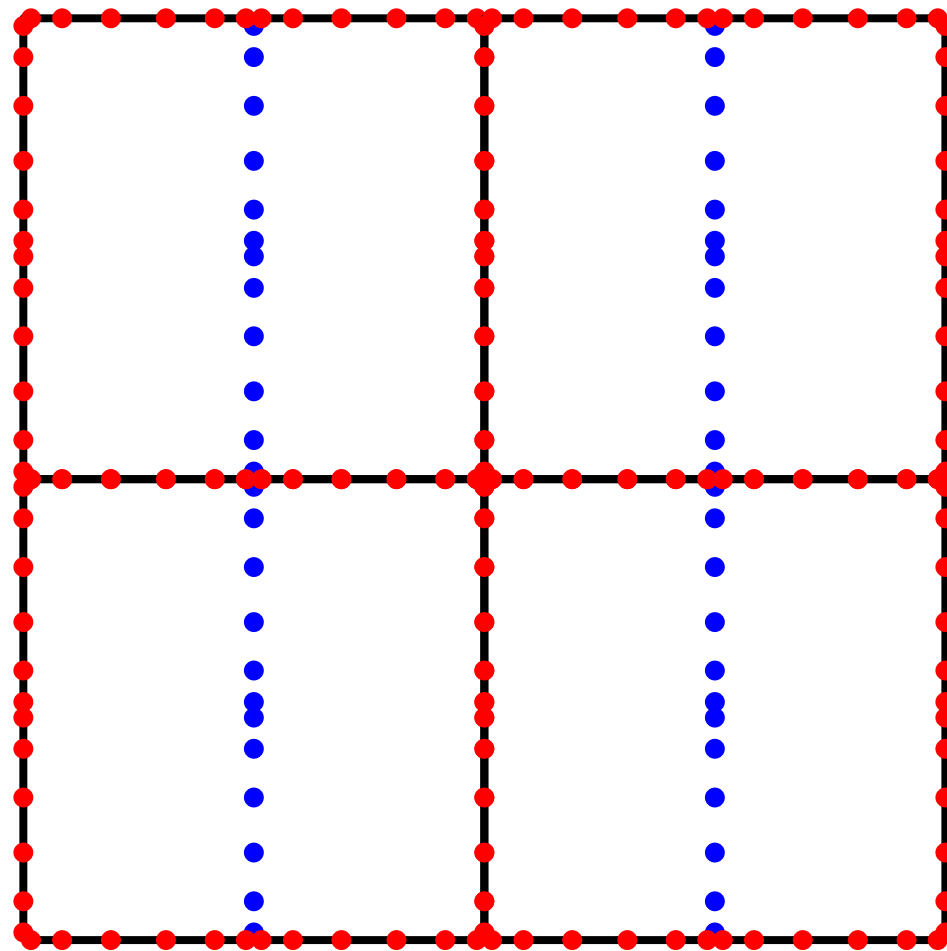
Model problem: Given f , g , and b , find u such that

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

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

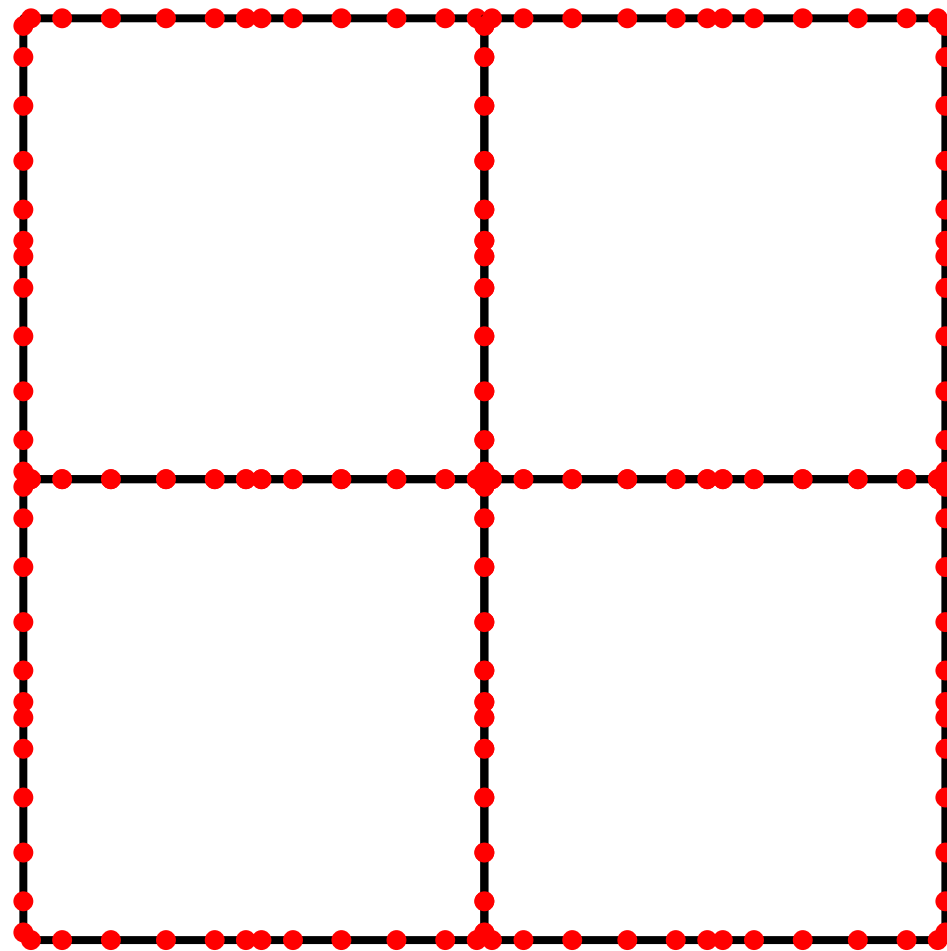
Model problem: Given f , g , and b , find u such that

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

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

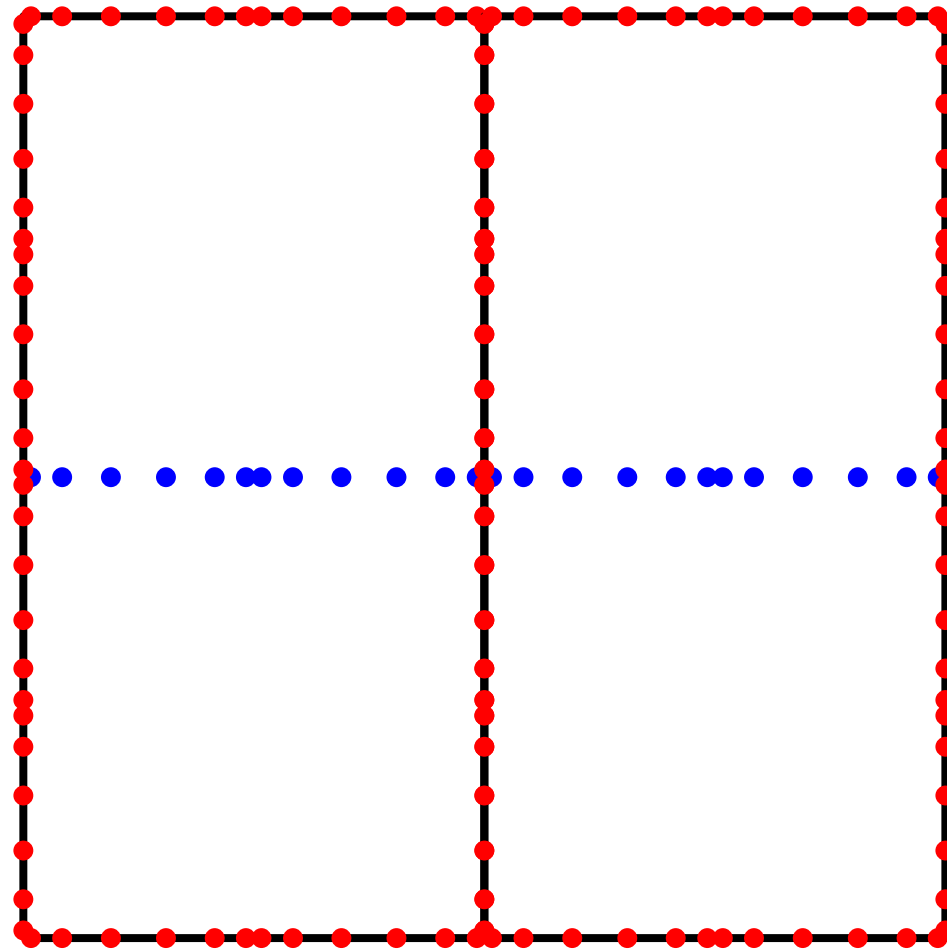
Model problem: Given f , g , and b , find u such that

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

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

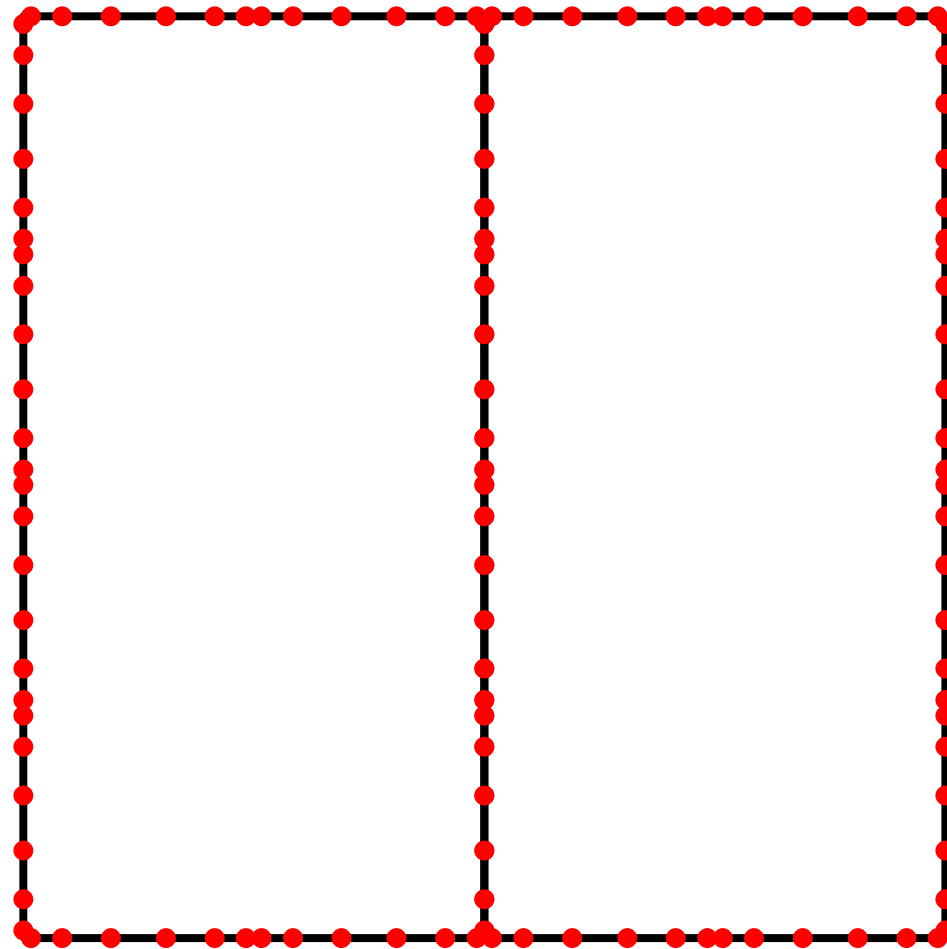
Model problem: Given f , g , and b , find u such that

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

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

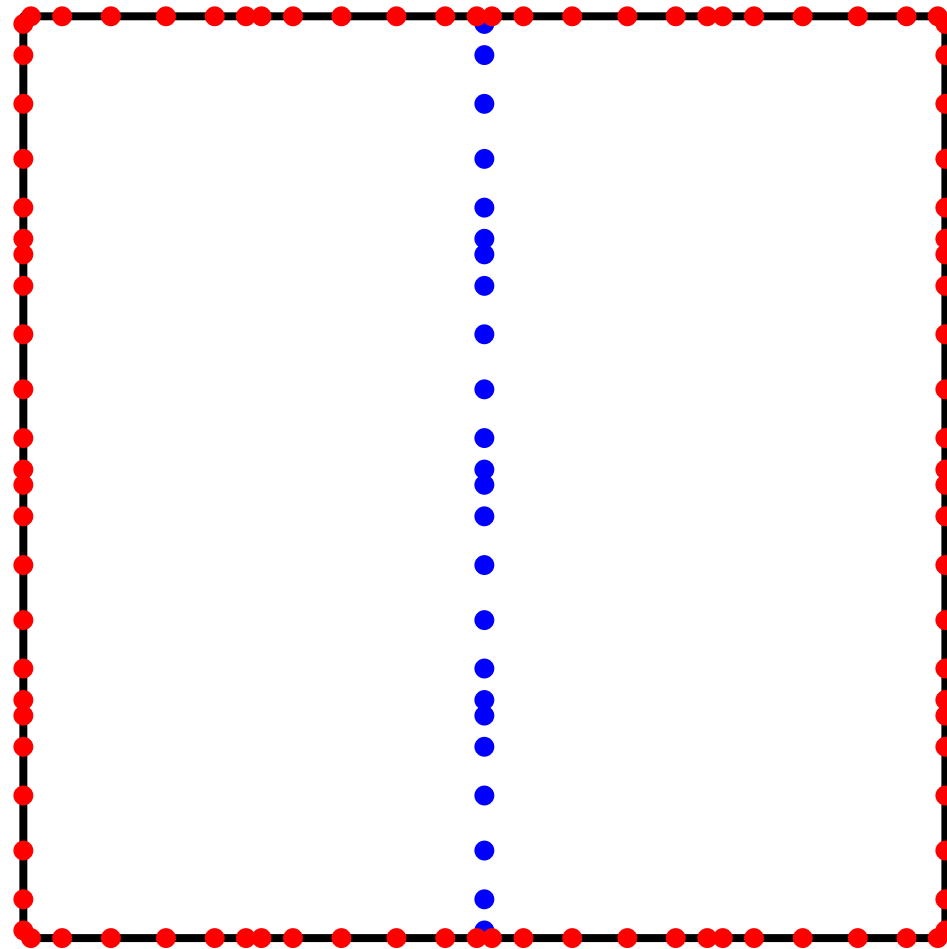
Model problem: Given f , g , and b , find u such that

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

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

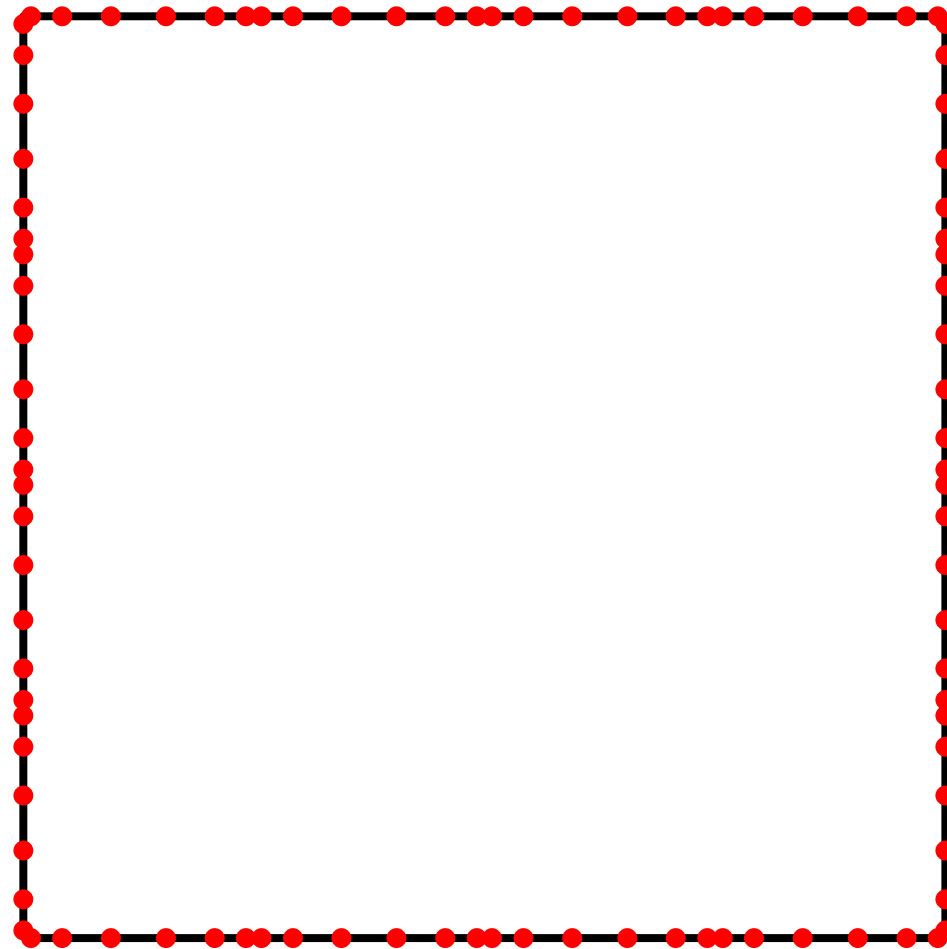
Model problem: Given f , g , and b , find u such that

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

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.



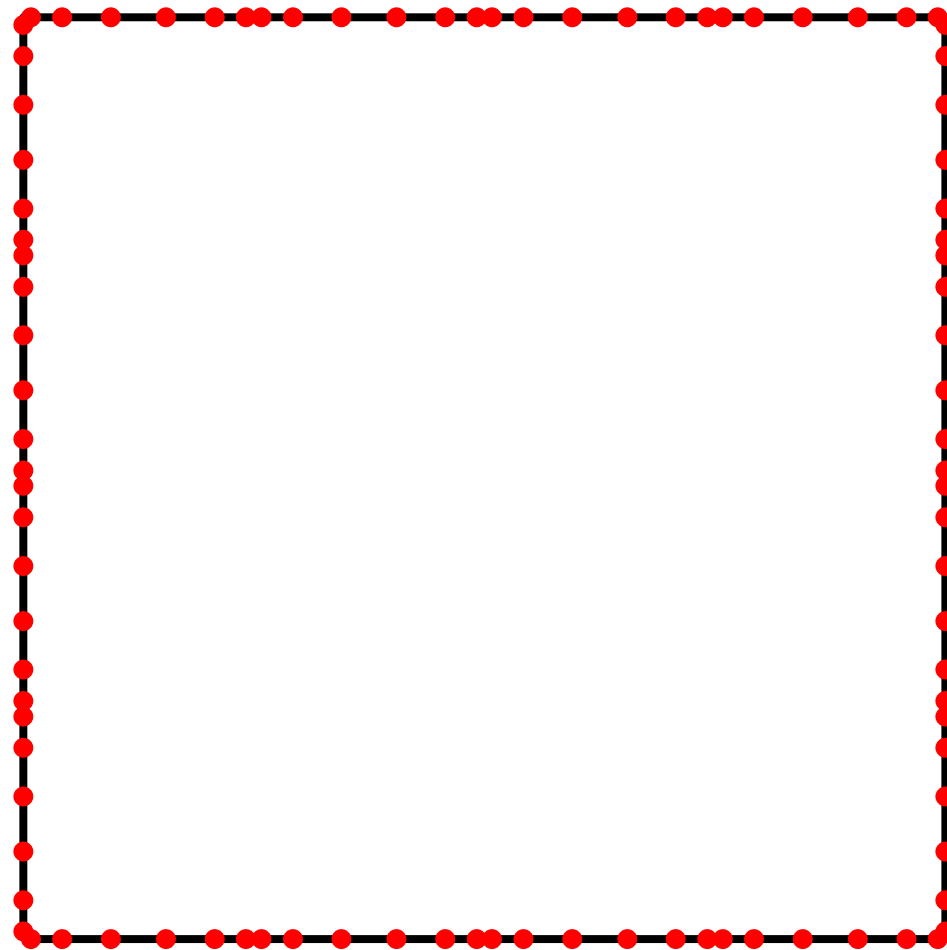
The Hierarchical Poincaré-Steklov Method

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

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

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.



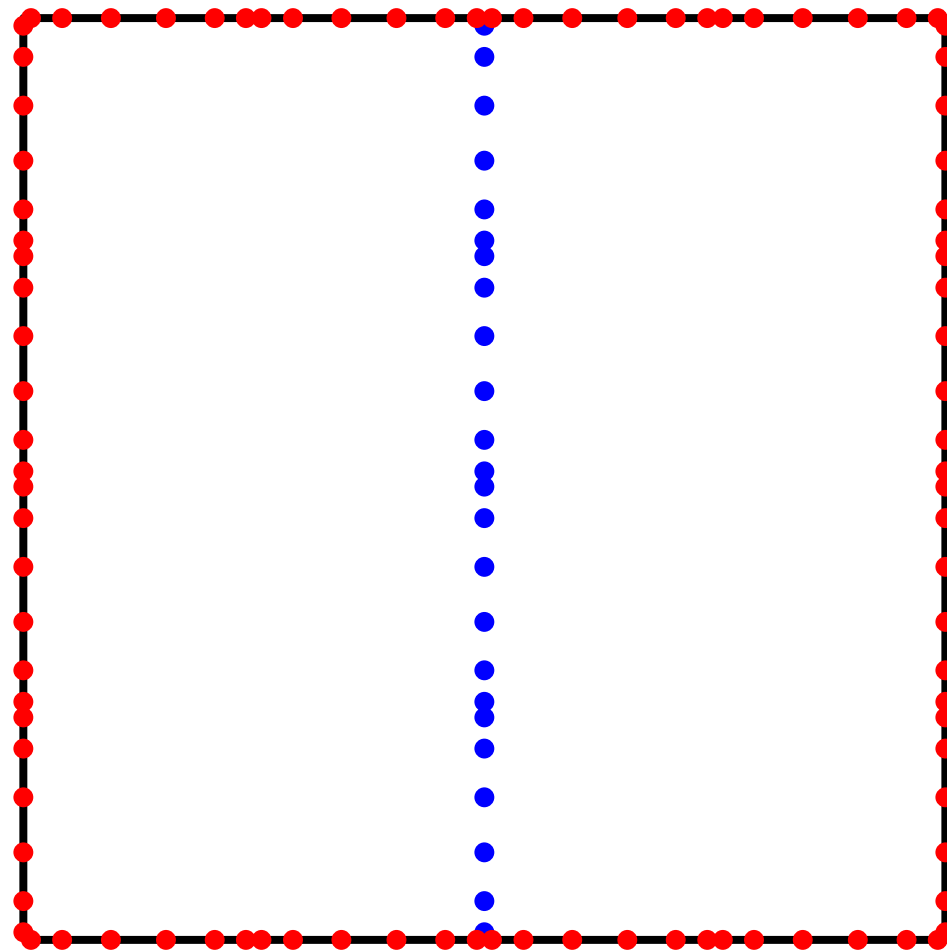
The Hierarchical Poincaré-Steklov Method

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

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

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.



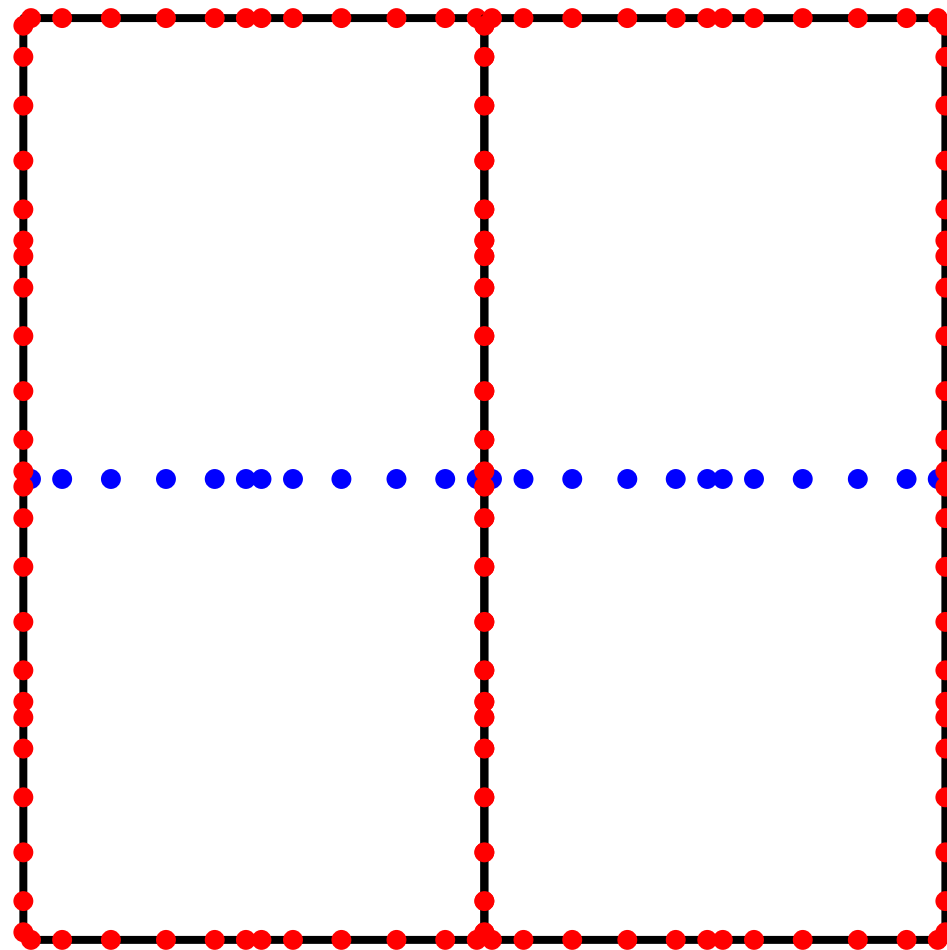
The Hierarchical Poincaré-Steklov Method

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

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

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.



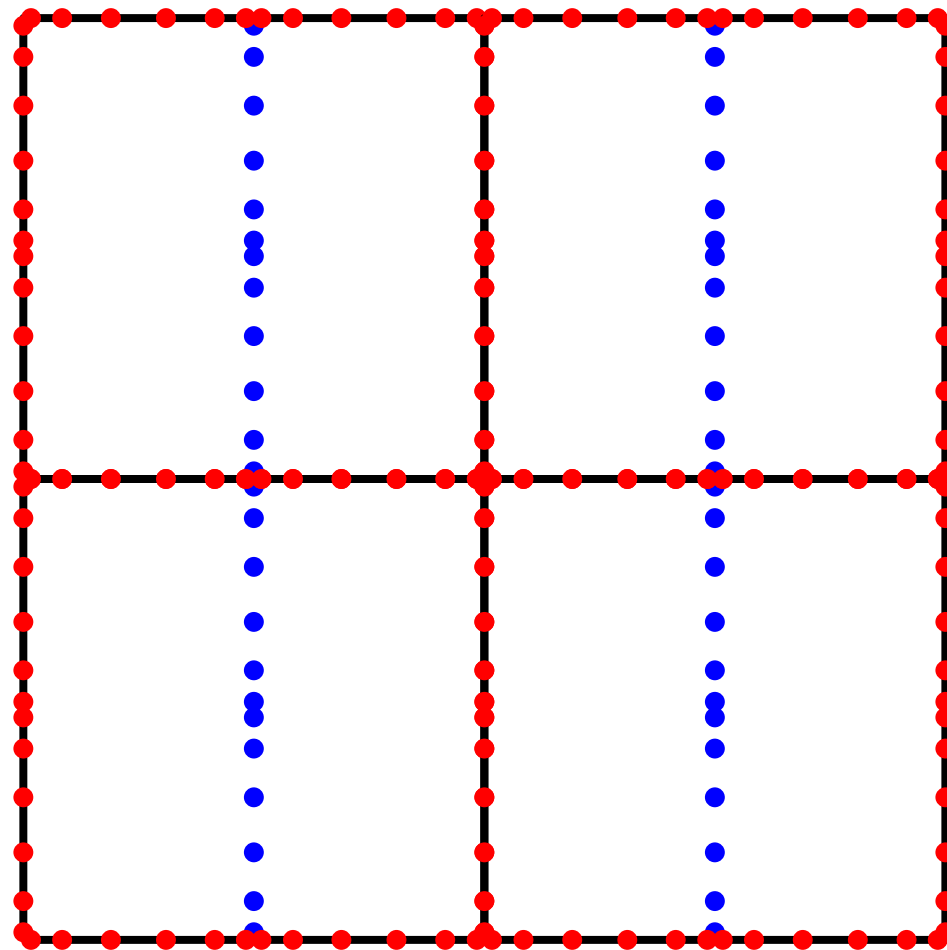
The Hierarchical Poincaré-Steklov Method

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

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

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.



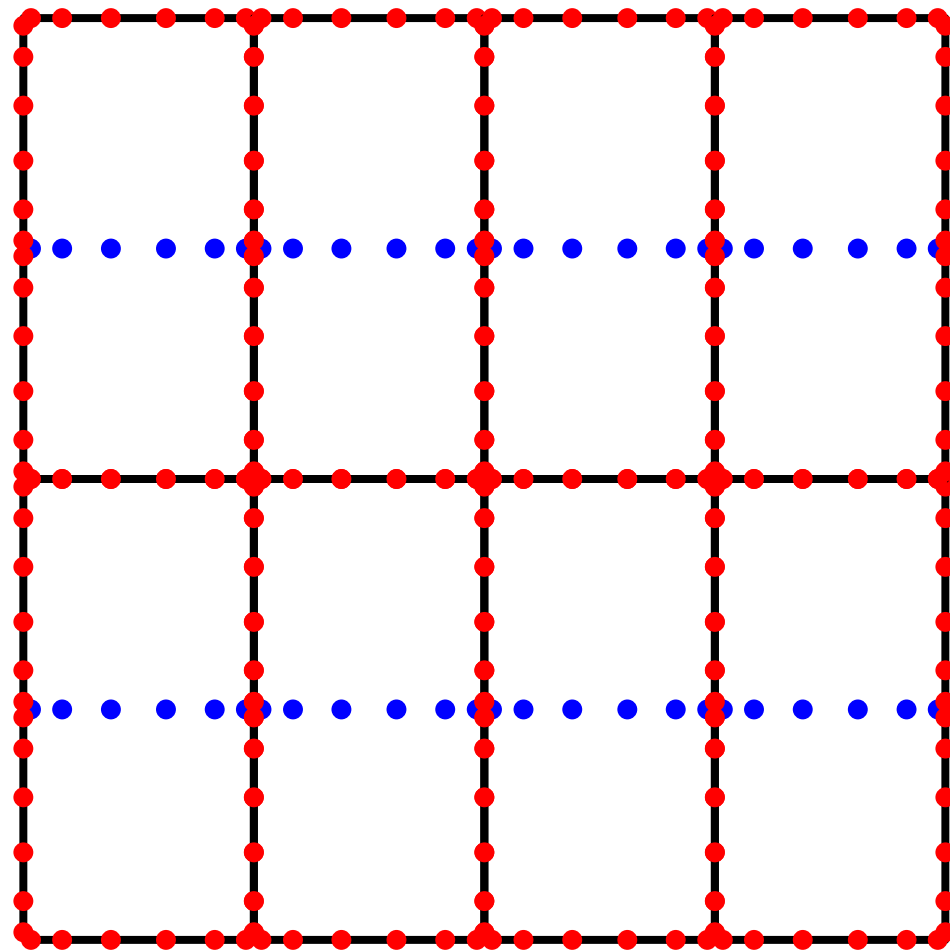
The Hierarchical Poincaré-Steklov Method

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

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

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.



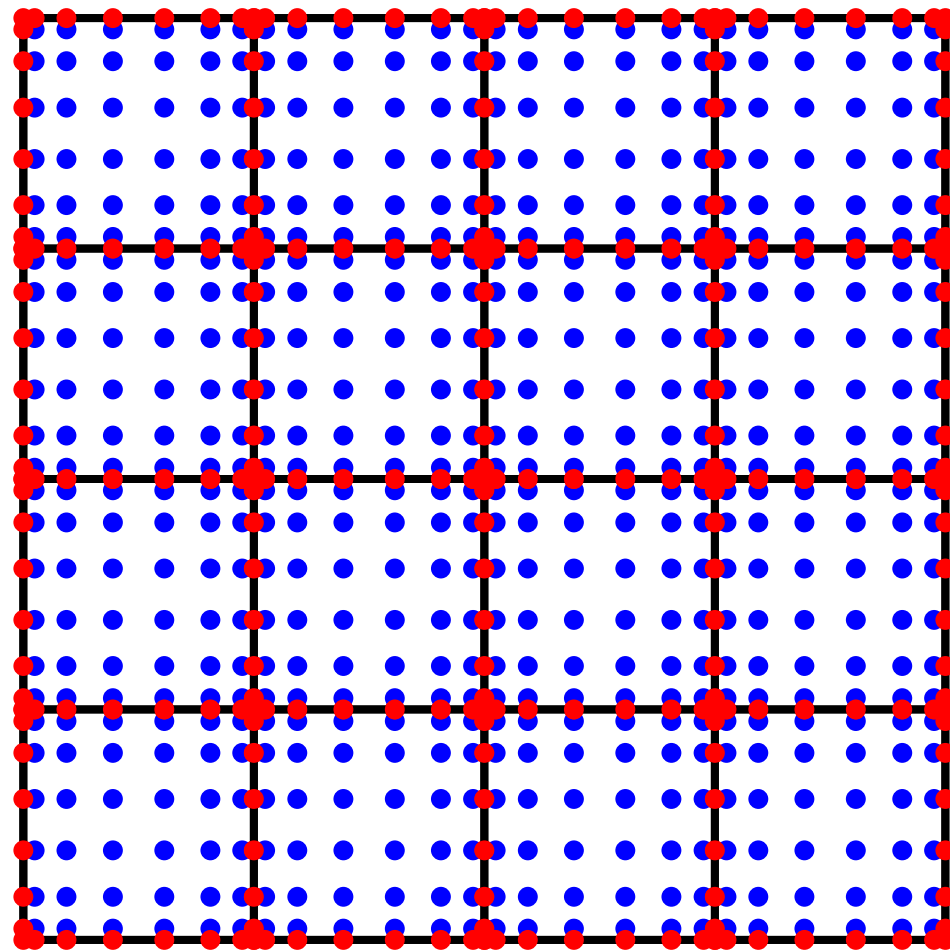
The Hierarchical Poincaré-Steklov Method

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

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

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:



- Joint work with Adrianna Gillman.
- In contrast to prior schemes, the speed of the solver does *not* deteriorate as the order is increased.
- Very high order can be used (say 20×20 local mesh).
 - Overall errors close to machine precision for problems with smooth solutions.
- Capable of solving 2D problems on domains that are several hundred wave-lengths across in minutes on a laptop.
- Exploiting internal structure in the boundary-to-boundary operators, we can attain $O(N)$ complexity and handle $N \sim 10^8$ on a desktop. (For 2D problems; 3D is harder.)
- Extension to 3D is under way. (Joint work with A. Yesypenko and A. Gillman.)

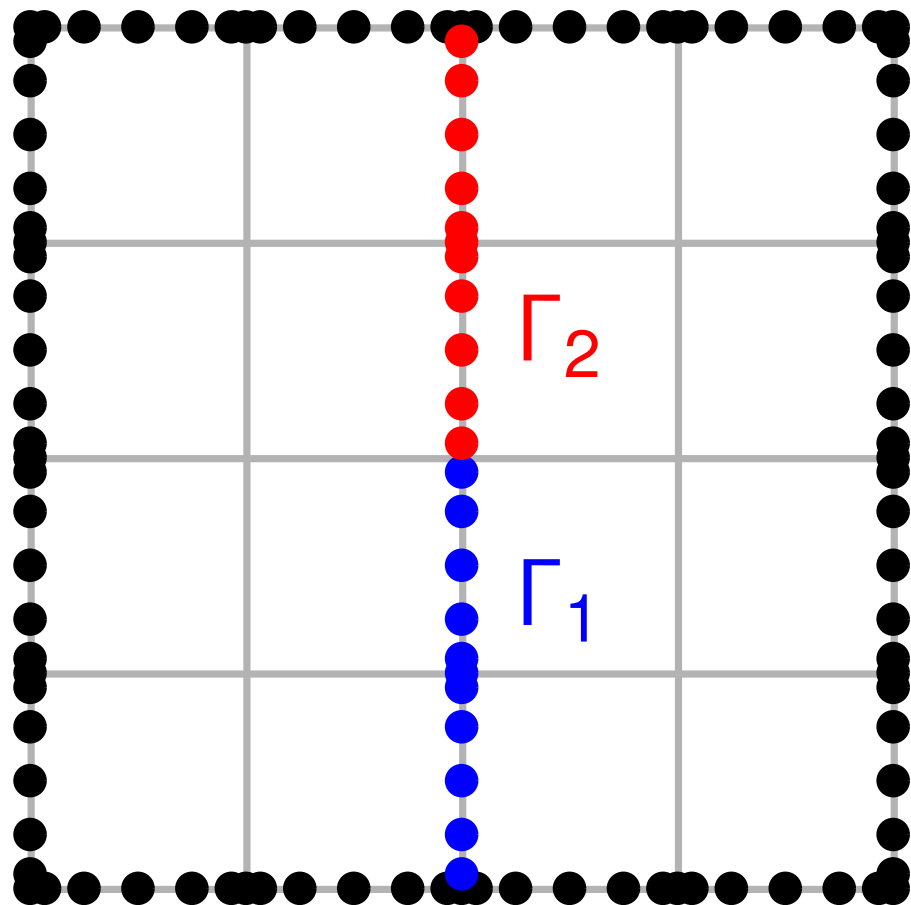
Hierarchical Poincaré-Steklov Method — rank deficiencies in the DtN operator:

Recall that at the top level, we need to invert a dense matrix that is defined on the nodes of the interface high-lighted in red and blue below. This matrix holds restrictions of the Dirichlet-to-Neumann (DtN) operators for the two blocks. We have claimed that this matrix is rank-structured. *But what are the ranks?*

Hierarchical Poincaré-Steklov Method — rank deficiencies in the DtN operator:

Recall that at the top level, we need to invert a dense matrix that is defined on the nodes of the interface high-lighted in red and blue below. This matrix holds restrictions of the Dirichlet-to-Neumann (DtN) operators for the two blocks. We have claimed that this matrix is rank-structured. *But what are the ranks?*

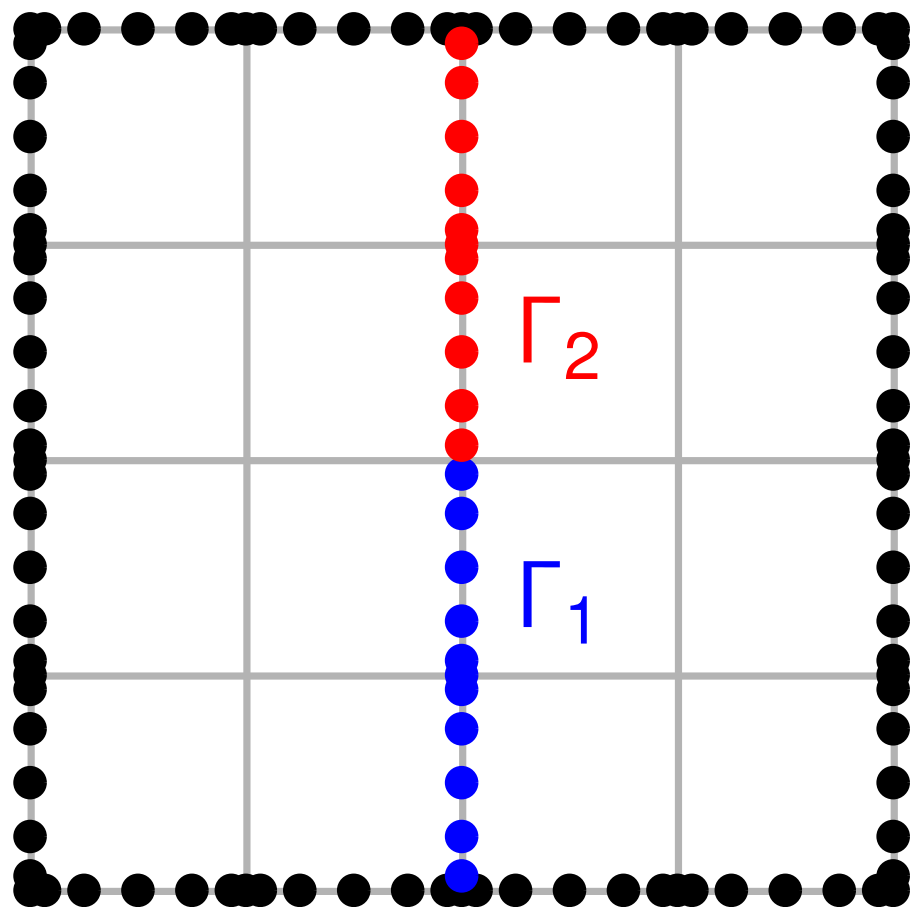
Let \mathbf{T} denote the restriction of the DtN matrix mapping Dirichlet data on Γ_1 to Neumann data on Γ_2 for a $1\,089 \times 1\,089$ grid. Then \mathbf{T} is of size 512×512 .



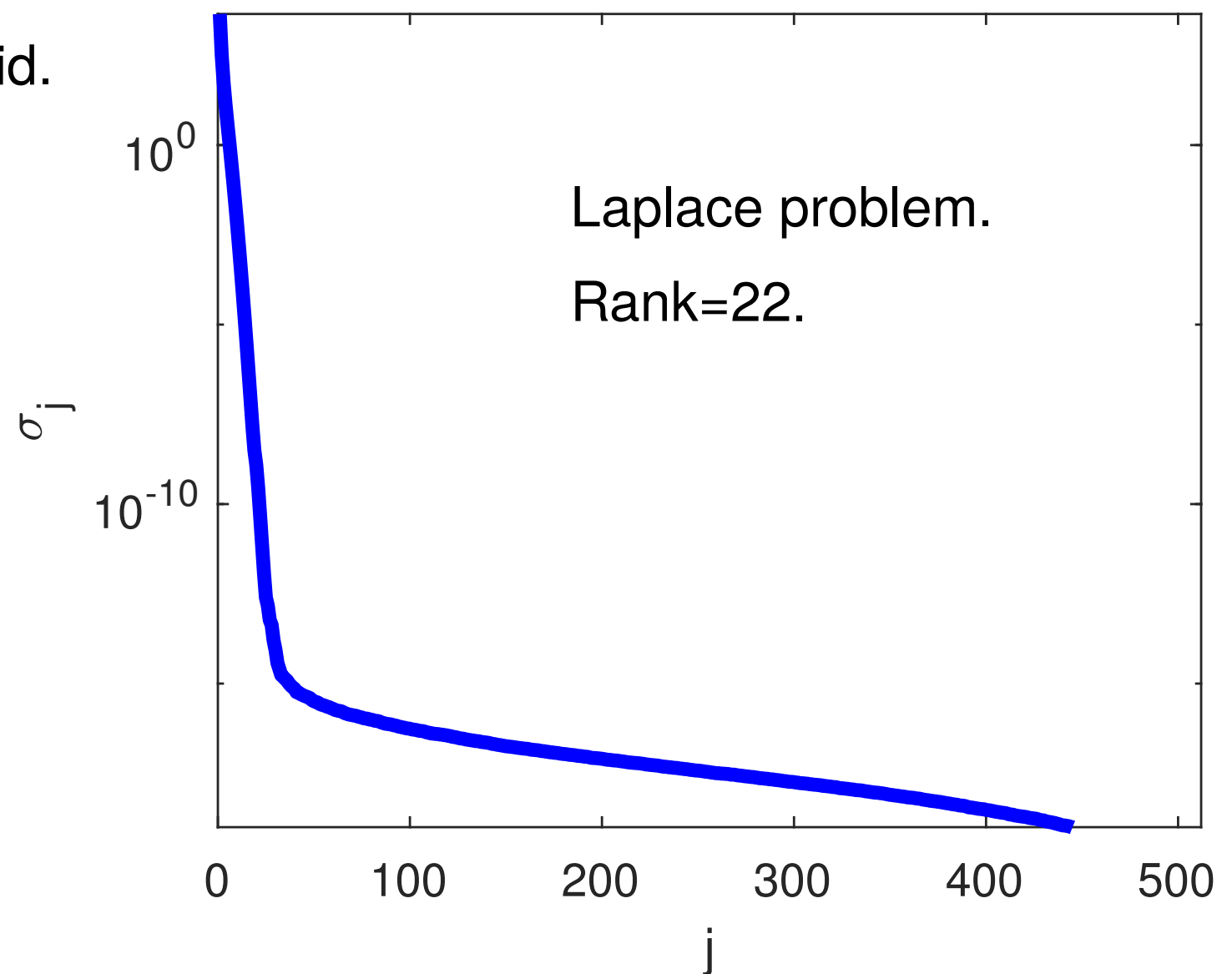
Hierarchical Poincaré-Steklov Method — rank deficiencies in the DtN operator:

Recall that at the top level, we need to invert a dense matrix that is defined on the nodes of the interface high-lighted in red and blue below. This matrix holds restrictions of the Dirichlet-to-Neumann (DtN) operators for the two blocks. We have claimed that this matrix is rank-structured. *But what are the ranks?*

Let \mathbf{T} denote the restriction of the DtN matrix mapping Dirichlet data on Γ_1 to Neumann data on Γ_2 for a $1\,089 \times 1\,089$ grid. Then \mathbf{T} is of size 512×512 .



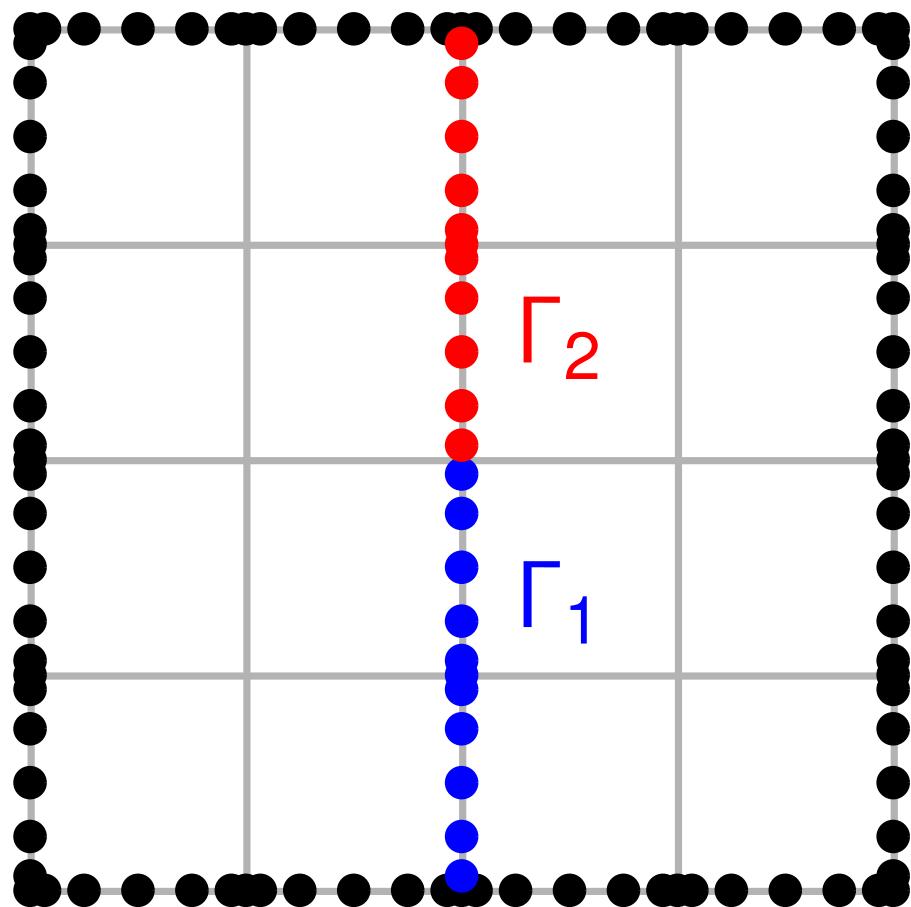
Singular values of \mathbf{T} .



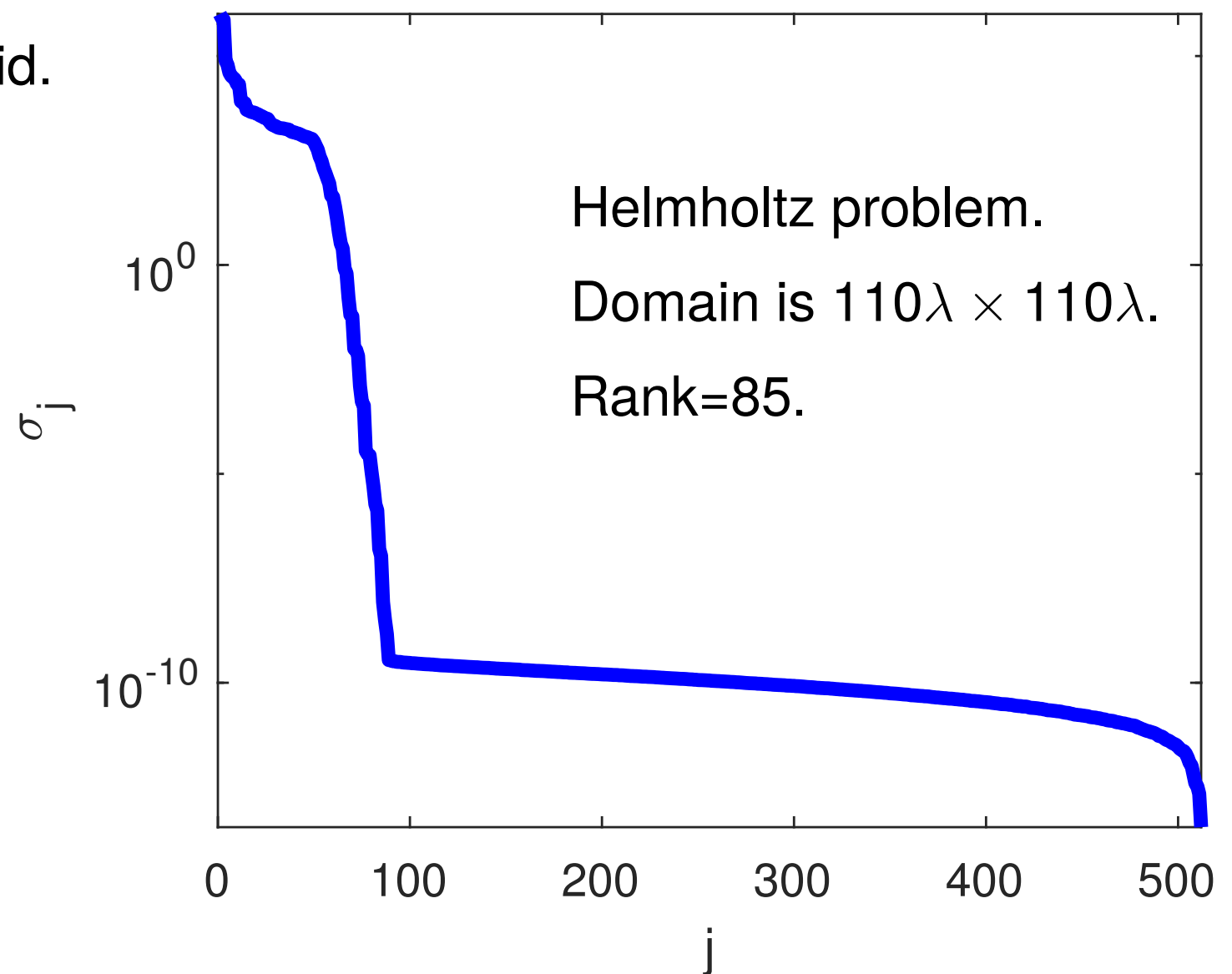
Hierarchical Poincaré-Steklov Method — rank deficiencies in the DtN operator:

Recall that at the top level, we need to invert a dense matrix that is defined on the nodes of the interface high-lighted in red and blue below. This matrix holds restrictions of the Dirichlet-to-Neumann (DtN) operators for the two blocks. We have claimed that this matrix is rank-structured. *But what are the ranks?*

Let \mathbf{T} denote the restriction of the DtN matrix mapping Dirichlet data on Γ_1 to Neumann data on Γ_2 for a $1\,089 \times 1\,089$ grid. Then \mathbf{T} is of size 512×512 .



Singular values of \mathbf{T} .



Hierarchical Poincaré-Steklov Method: numerical results

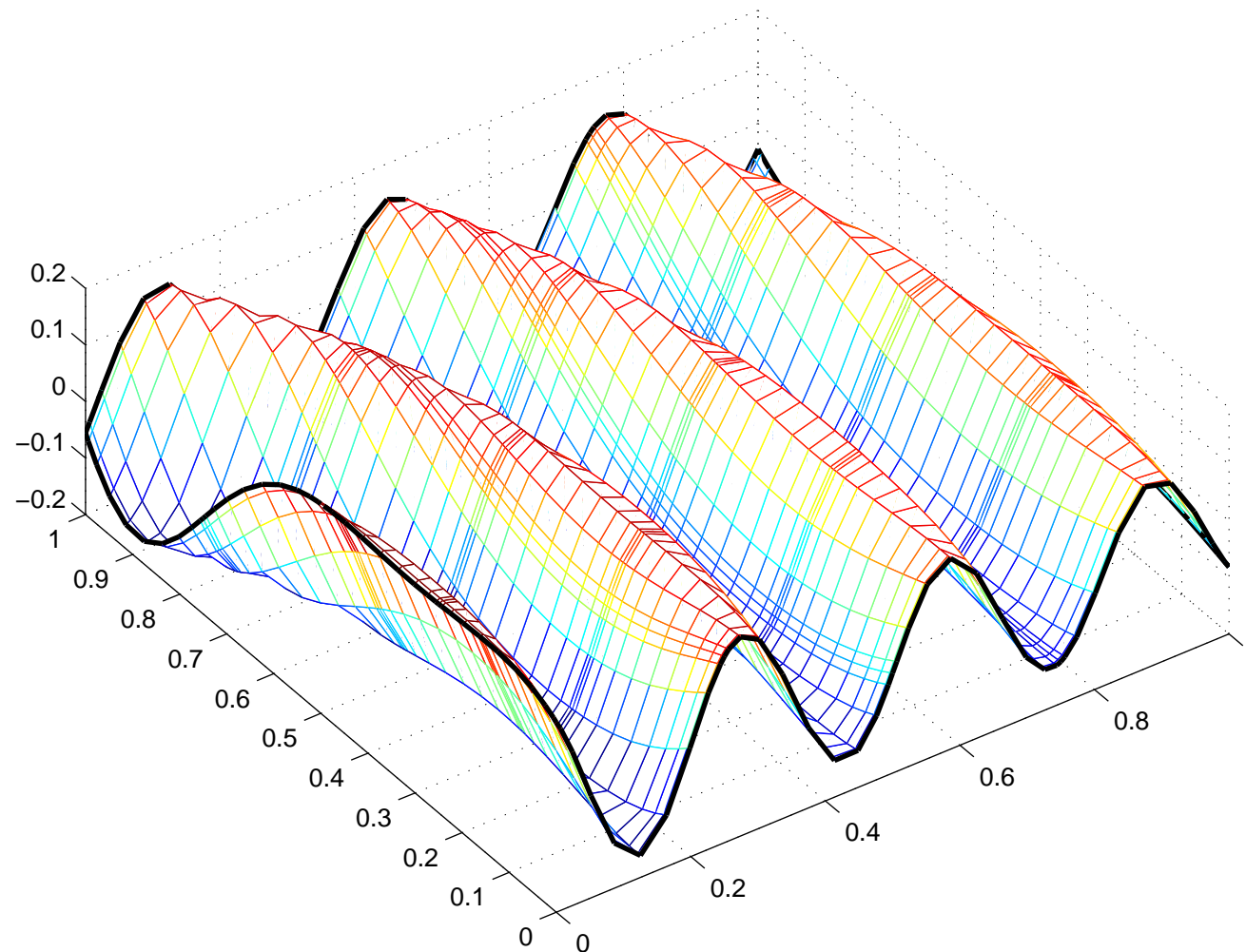
Set $\Omega = [0, 1]^2$ and $\Gamma = \partial\Omega$. Consider the problem

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

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

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

Approximate solution. ntot=1681 pts-per-wave=12.00



Hierarchical Poincaré-Steklov Method: numerical results

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

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

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

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

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: $e = \max_{\mathbf{x} \in \Omega} |u(\mathbf{x}) - u_{\text{exact}}(\mathbf{x})|$.

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

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

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

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

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

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

ρ	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

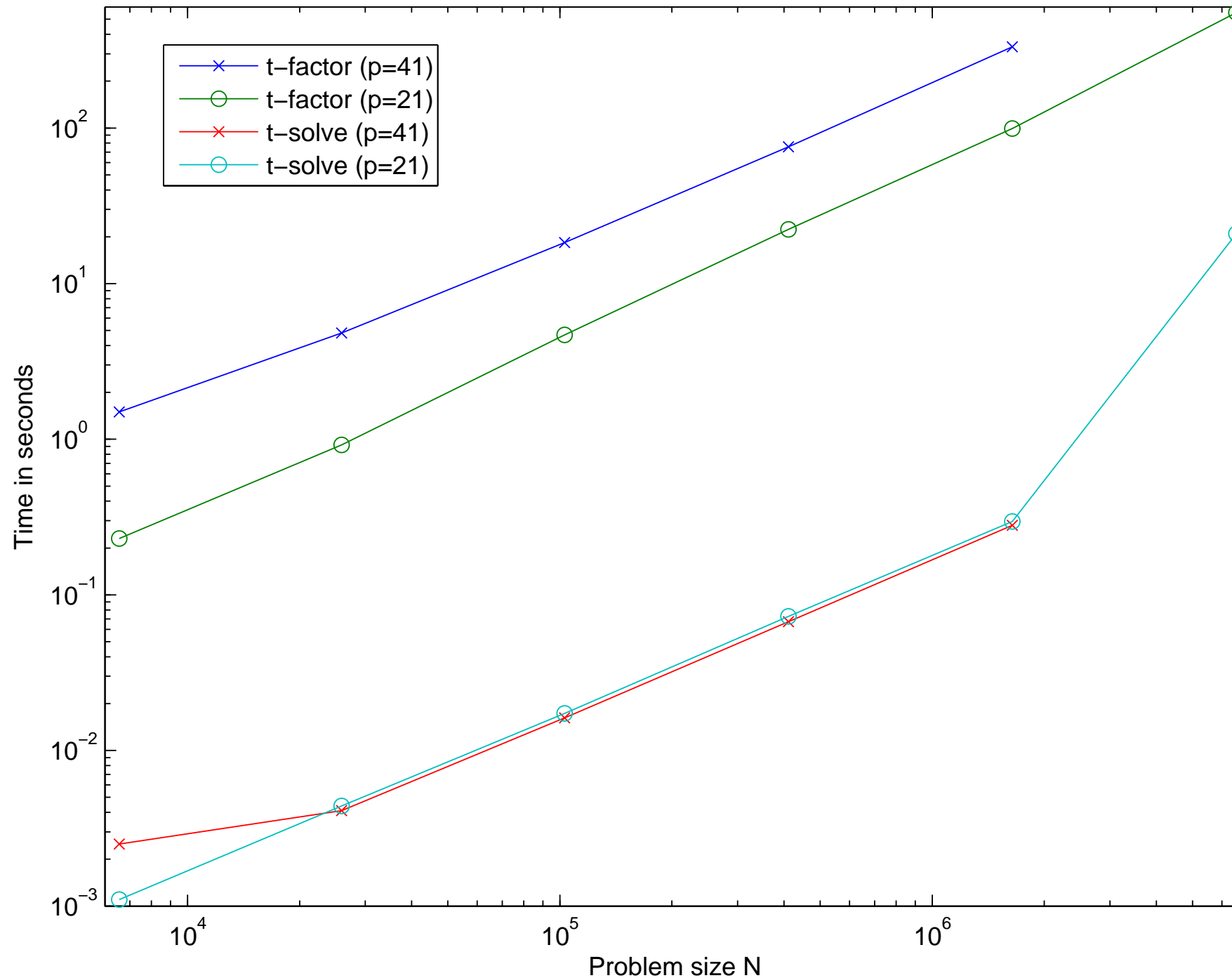
Error is measured in sup-norm: $e = \max_{\mathbf{x} \in \Omega} |u(\mathbf{x}) - u_{\text{exact}}(\mathbf{x})|$.

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

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

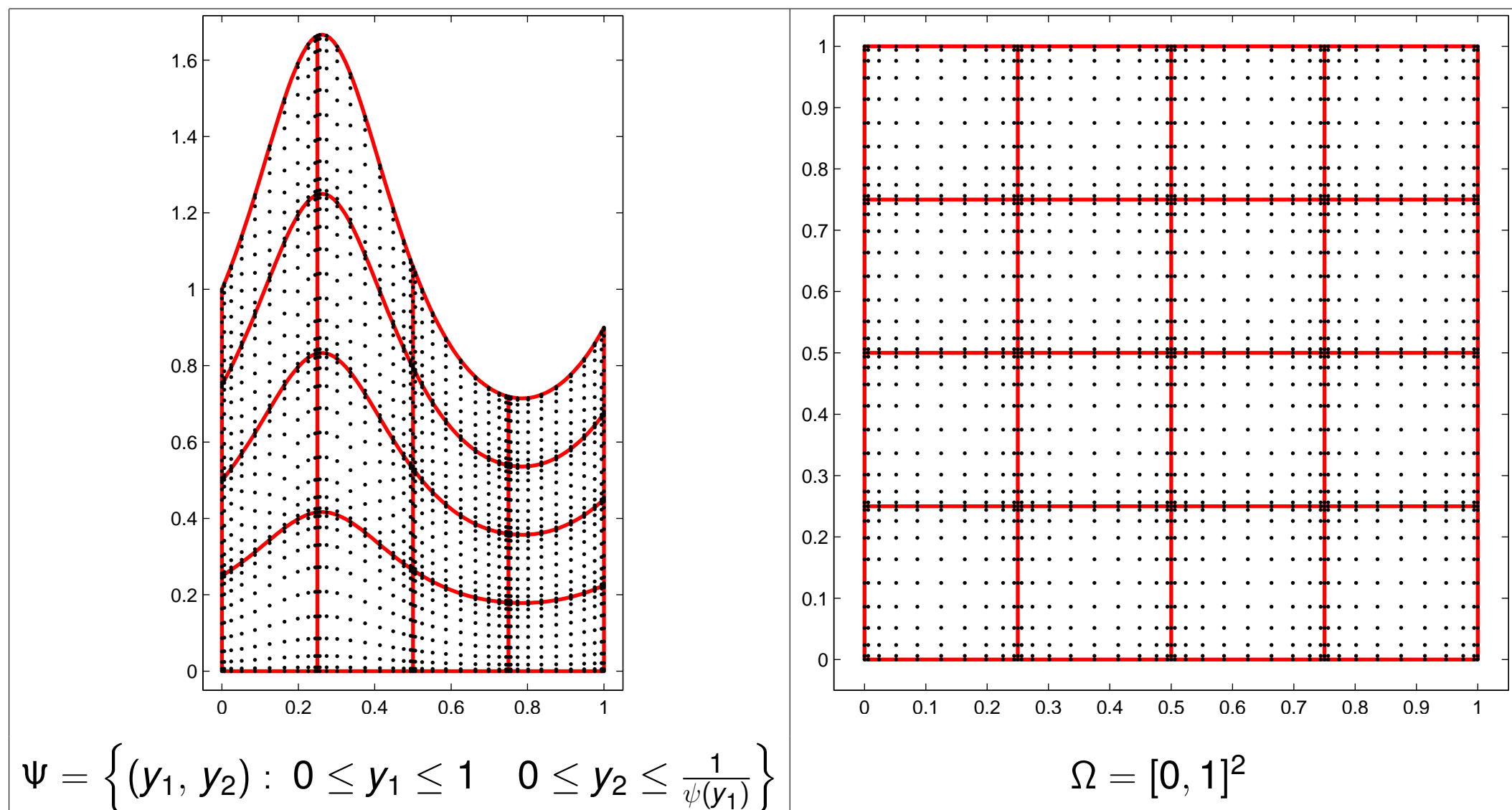
where $\Omega = [0, 1]^2$, where $\Gamma = \partial\Omega$, and where $b(\mathbf{x}) = (\sin(4\pi x_1) \sin(4\pi x_2))^2$.

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 $f(\mathbf{x}) = \cos(8x_1) (1 - 2x_2)$.

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

ρ	N	pts per wave	$u_N(\hat{\mathbf{x}})$	E_N^{int}	$w_N(\hat{\mathbf{y}})$	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



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

$$(9) \quad \begin{cases} -\Delta u(\mathbf{y}) - \kappa^2 u(\mathbf{y}) = 0 & \mathbf{y} \in \Psi, \\ u(\mathbf{y}) = f(\mathbf{y}) & \mathbf{y} \in \partial\Psi. \end{cases}$$

The reparameterization is $y_1 = x_1$ and $y_2 = \psi(y_1)y_2$, and so the Helmholtz equation (9)

takes the form

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

Numerical results for curved domain

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

N	<i>Exact solution known</i>		<i>Dirichlet data $f = 1$</i>		
	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	—	—	—

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.

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*

$$T : u|_{\Gamma} \mapsto \frac{\partial u}{\partial n} \Big|_{\Gamma}$$

consider the *impedance map*

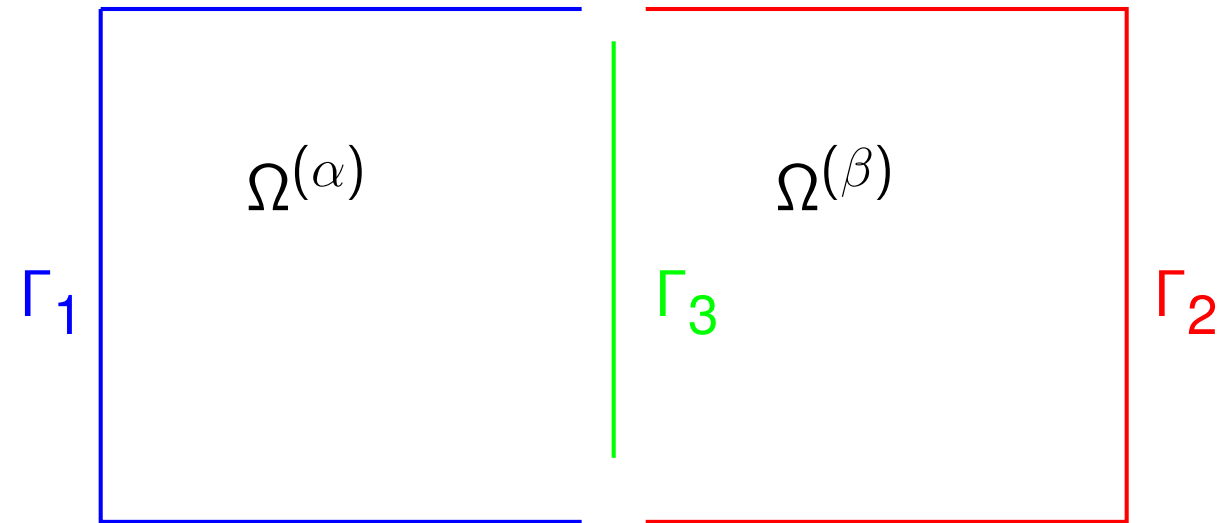
$$E : u|_{\Gamma} + i \frac{\partial u}{\partial n} \Big|_{\Gamma} \mapsto u|_{\Gamma} - i \frac{\partial u}{\partial n} \Big|_{\Gamma}$$

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 $\Gamma_1, \Gamma_2, \Gamma_3$:

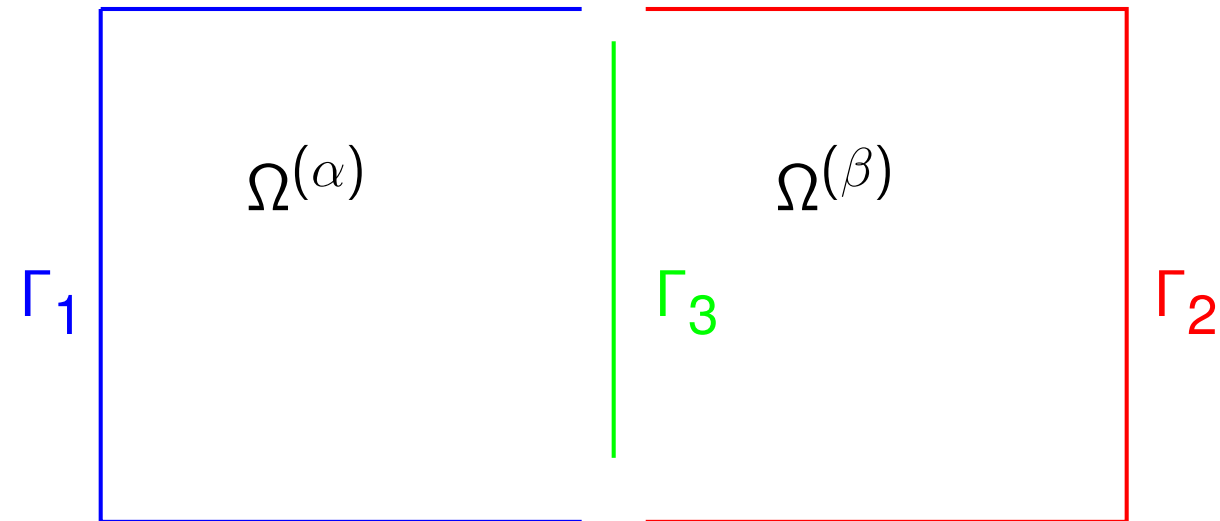


In the composite spectral method we have

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

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

Consider the merge of two patches $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ with boundaries $\Gamma_1, \Gamma_2, \Gamma_3$:



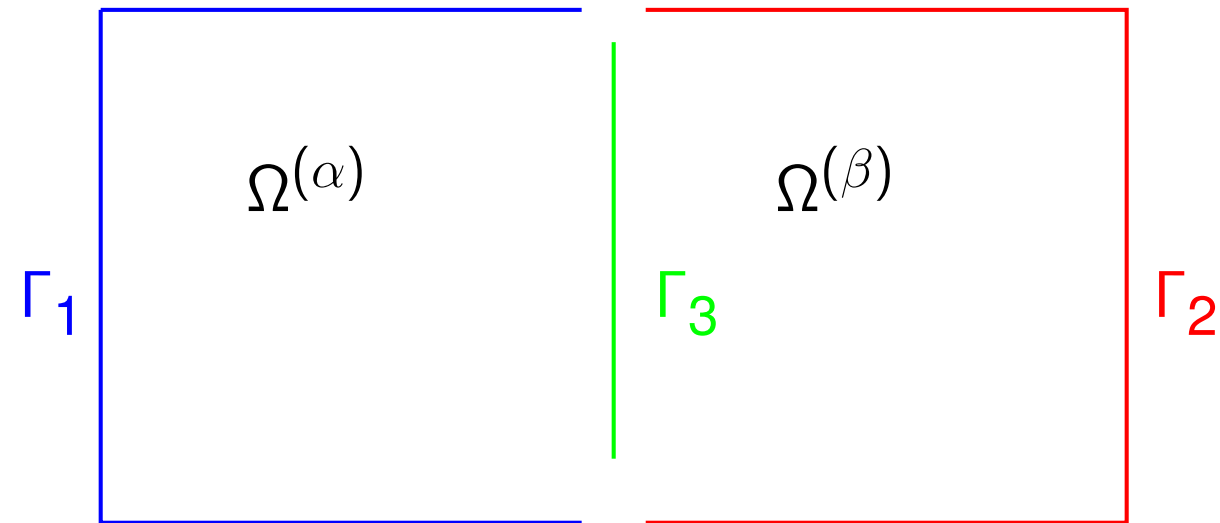
In the composite spectral method we have

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

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 $\Gamma_1, \Gamma_2, \Gamma_3$:



In the composite spectral method we have

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

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

$$\begin{array}{ccccc} \mathbf{A}_{12} & = & \mathbf{U}_1 & \tilde{\mathbf{A}}_{12} & \mathbf{V}_2^* \\ n \times n & & n \times k & k \times k & k \times n \end{array} \quad \text{and} \quad \begin{array}{ccccc} \mathbf{A}_{21} & = & \mathbf{U}_2 & \tilde{\mathbf{A}}_{21} & \mathbf{V}_1^* \\ n \times n & & n \times k & k \times k & k \times n \end{array}$$

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

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

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

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

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

$$(10) \quad \begin{cases} -\Delta u(\mathbf{x}) - c_1(\mathbf{x}) \partial_1 u(\mathbf{x}) - c_2(\mathbf{x}) \partial_2 u(\mathbf{x}) - c(\mathbf{x}) u(\mathbf{x}) = 0, & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases}$$

Laplace Let $c_1(\mathbf{x}) = c_2(\mathbf{x}) = c(\mathbf{x}) = 0$ in (10).

Helmholtz I Let $c_1(\mathbf{x}) = c_2(\mathbf{x}) = 0$, and $c(\mathbf{x}) = \kappa^2$ where $\kappa = 80$ in (10). 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 $c_1(\mathbf{x}) = c_2(\mathbf{x}) = 0$, and $c(\mathbf{x}) = \kappa^2$ where $\kappa = 640$ in (10). This corresponds to a domain of size roughly 102×102 wave-lengths.

Helmholtz III We again set $c_1(\mathbf{x}) = c_2(\mathbf{x}) = 0$, and $c(\mathbf{x}) = \kappa^2$ in (10), 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

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

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

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

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

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

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

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

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:

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

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

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

Then for each time-step we need to solve

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

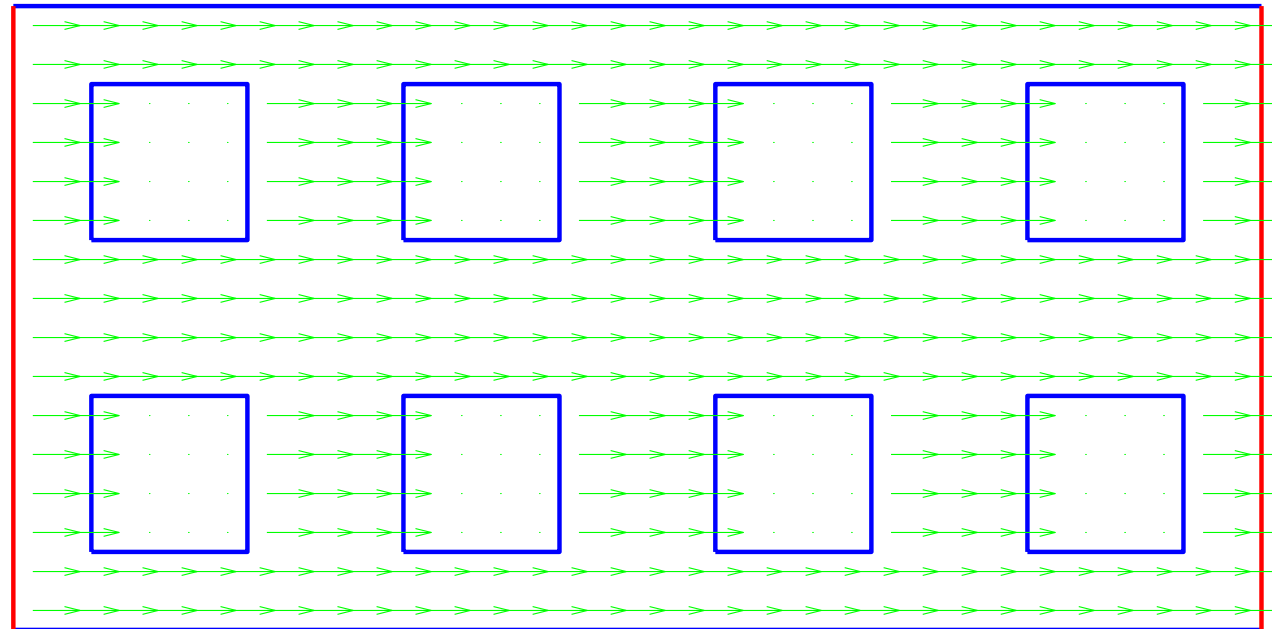
This is very well suited for our direct solver.

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*

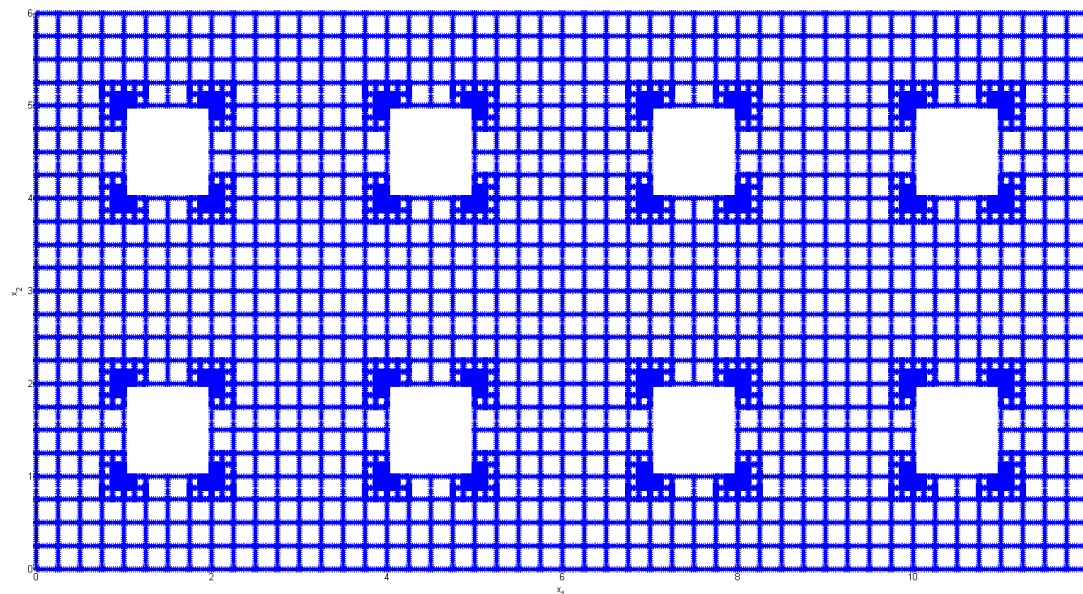
$$\frac{\partial u}{\partial t} - \Delta u + 30 \frac{\partial u}{\partial x_1} = 0,$$

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,

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

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

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

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

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

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

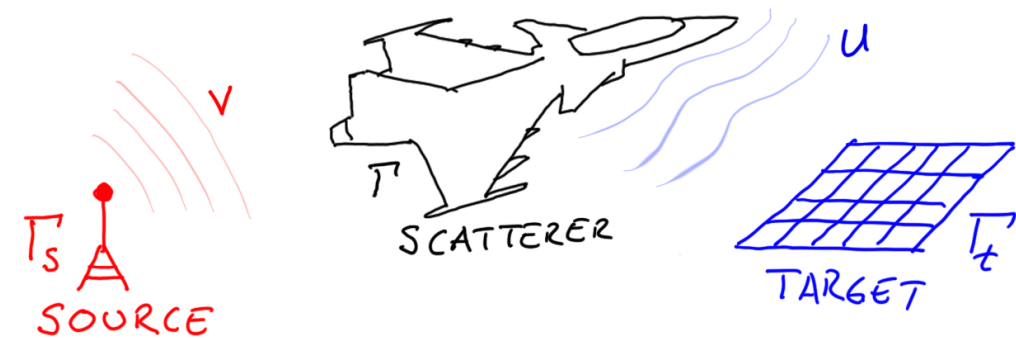
$$\exp(\tau B) \approx \sum_{m=-M}^M b_m (B - \alpha_m)^{-1}.$$

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.

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:



$$(11) \quad \begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 u(\mathbf{x}) = 0 & \mathbf{x} \in \mathbb{R}^3 \setminus \bar{\Omega} \\ u(\mathbf{x}) = v(\mathbf{x}) & \mathbf{x} \in \partial\Omega \\ \lim_{|\mathbf{x}| \rightarrow \infty} |\mathbf{x}| (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0. \end{cases}$$

The BVP (11) is in many ways equivalent to the BIE

$$(12) \quad -\pi i \sigma(\mathbf{x}) + \int_{\partial\Omega} \left(\left(\partial_{\mathbf{n}(\mathbf{y})} + i\kappa \right) \frac{e^{i\kappa|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|} \right) \sigma(\mathbf{y}) dS(\mathbf{y}) = f(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega.$$

The integral equation (12) has several advantages over the PDE (11), including:

- The domain of computation $\partial\Omega$ is finite.
- The domain of computation $\partial\Omega$ is 2D, while $\mathbb{R}^3 \setminus \bar{\Omega}$ is 3D.
- Equation (12) is inherently well-conditioned (as a “2nd kind Fredholm equation”).

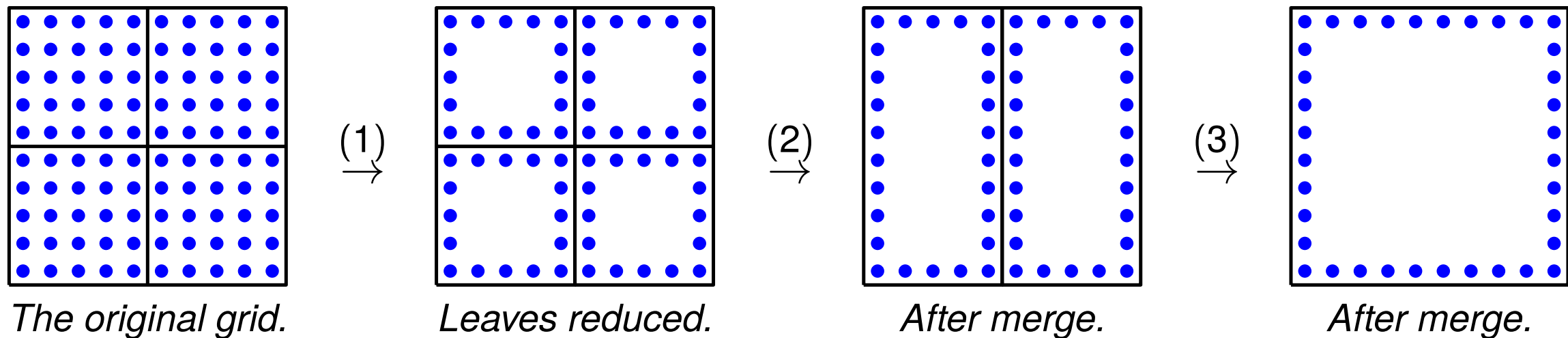
A serious drawback of integral equations is that they lead to *dense coefficient matrices*.

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

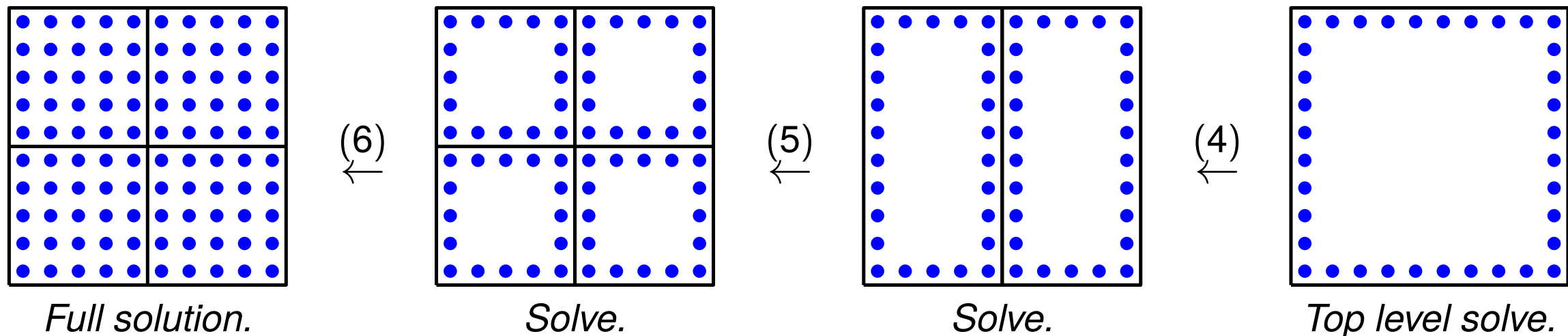
Direct solvers for integral equations

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

Upwards pass — build all solution operators:



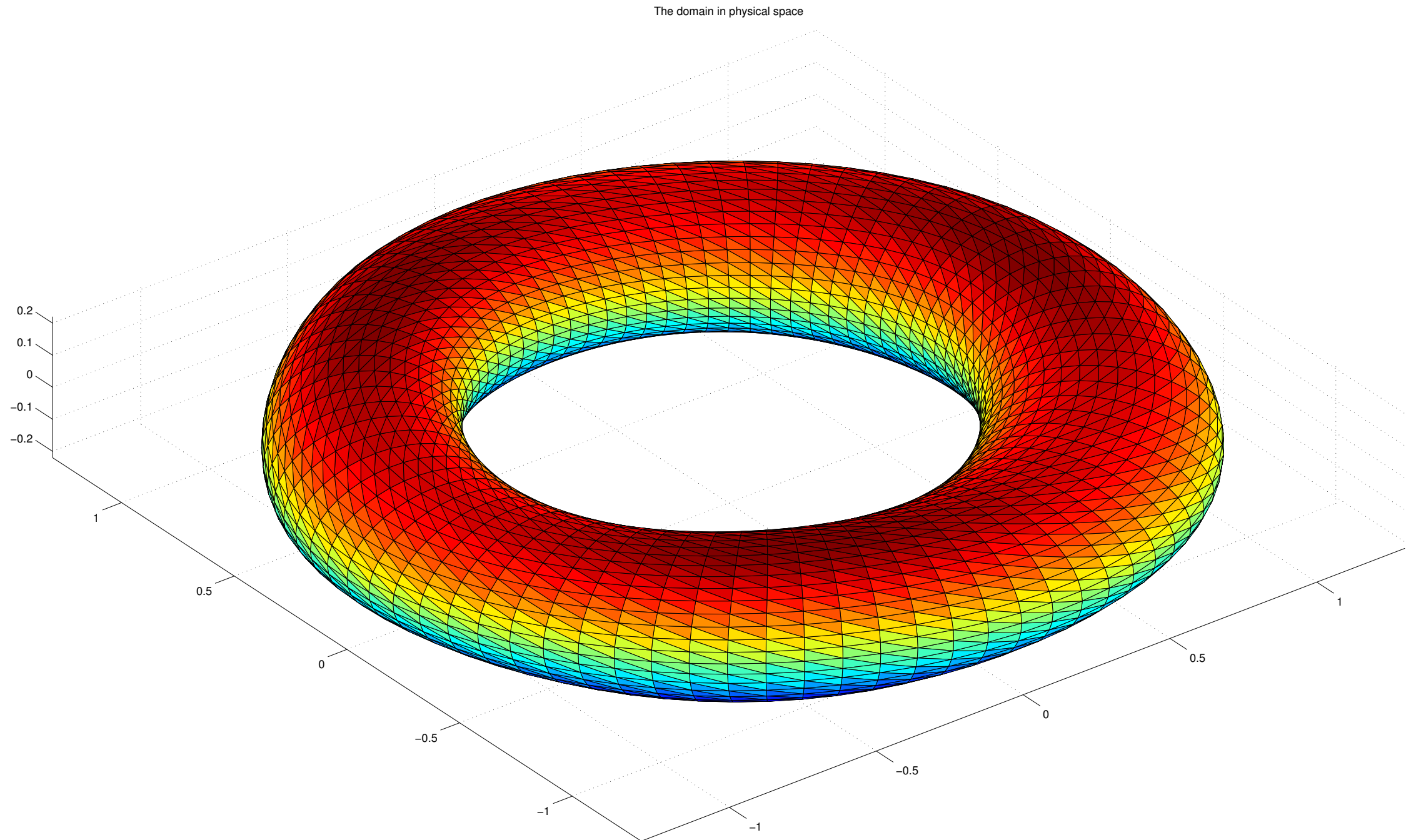
Downwards pass — solve for a particular data function (very fast!):



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 :

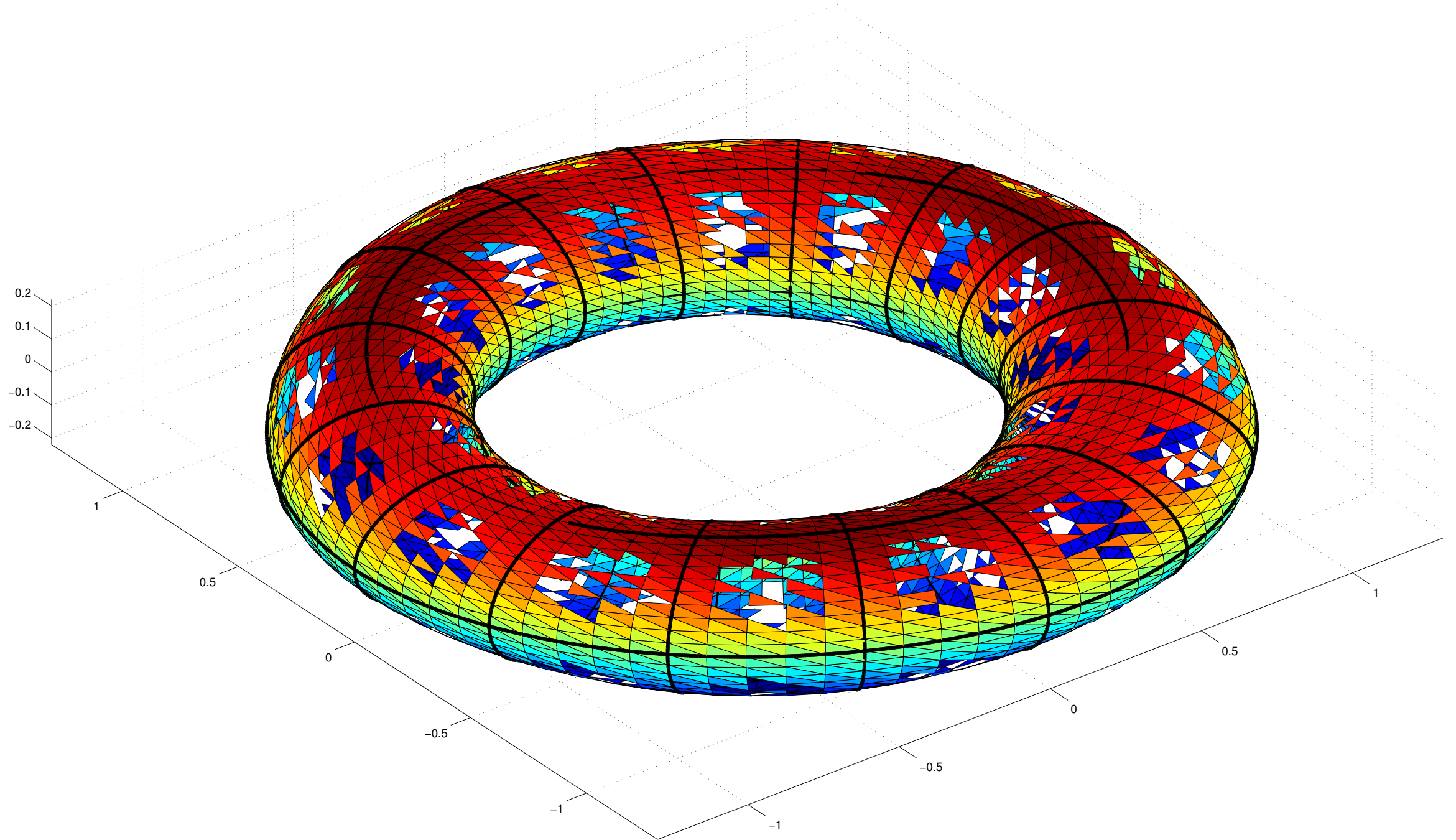


Let \mathbf{A} denote an $N \times N$ matrix arising upon discretizing a boundary integral operator

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

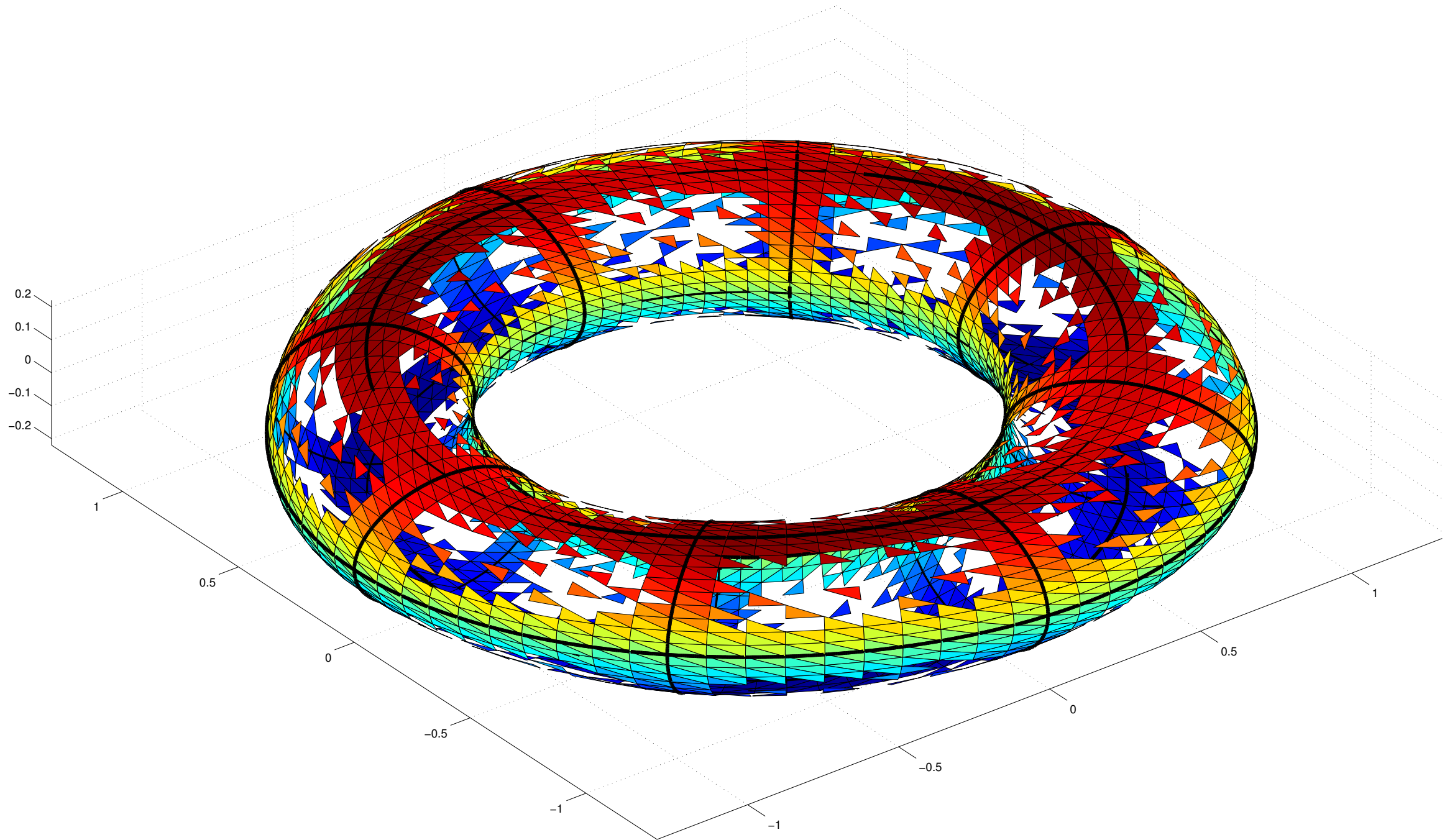
where Γ is the “torus-like” domain shown (it is deformed to avoid rotational symmetry).

The domain in physical space



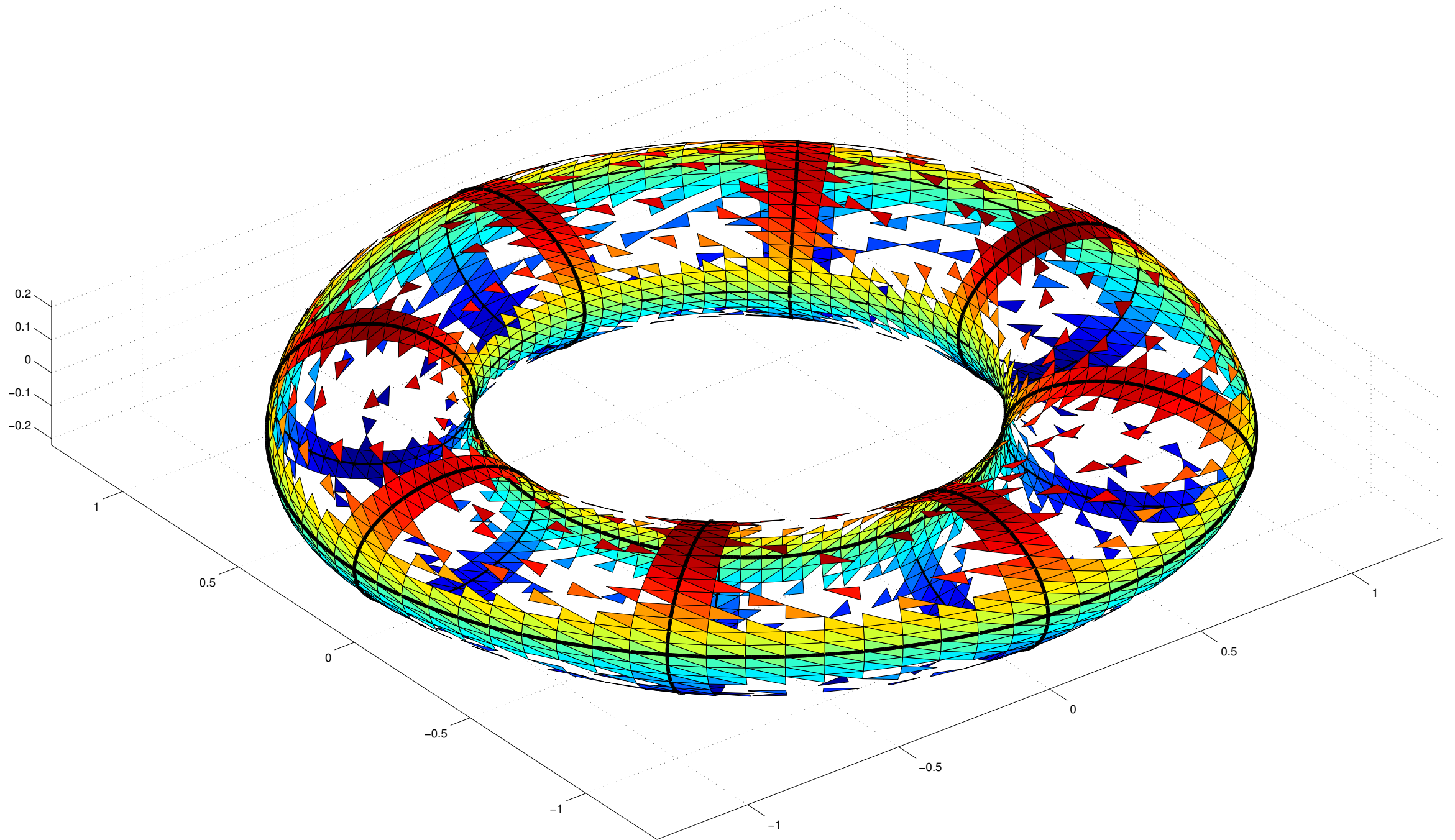
The reduced matrix represents a Nyström discretization supported on the panels shown.

The domain in physical space



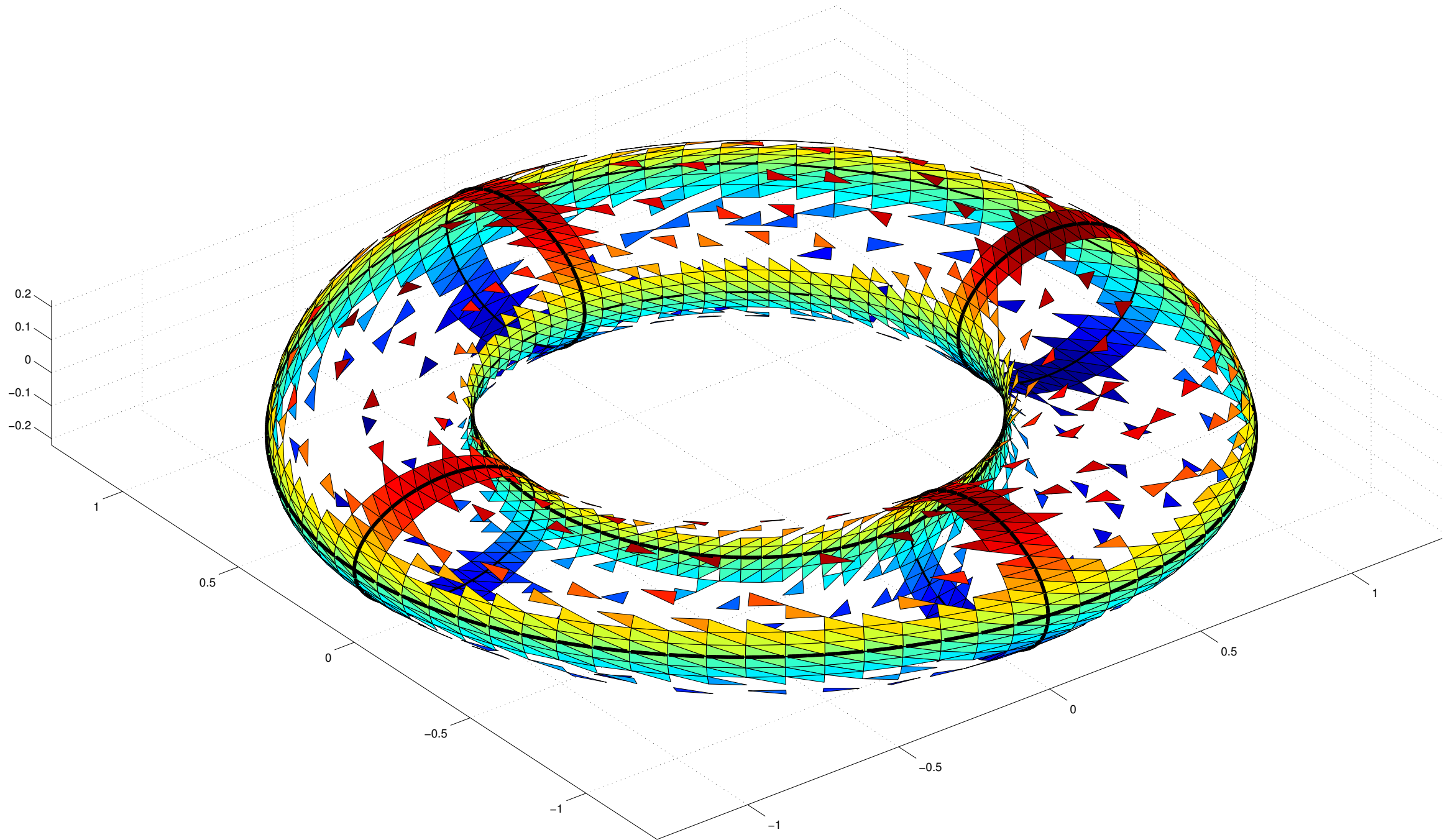
The reduced matrix represents a Nyström discretization supported on the panels shown.

The domain in physical space



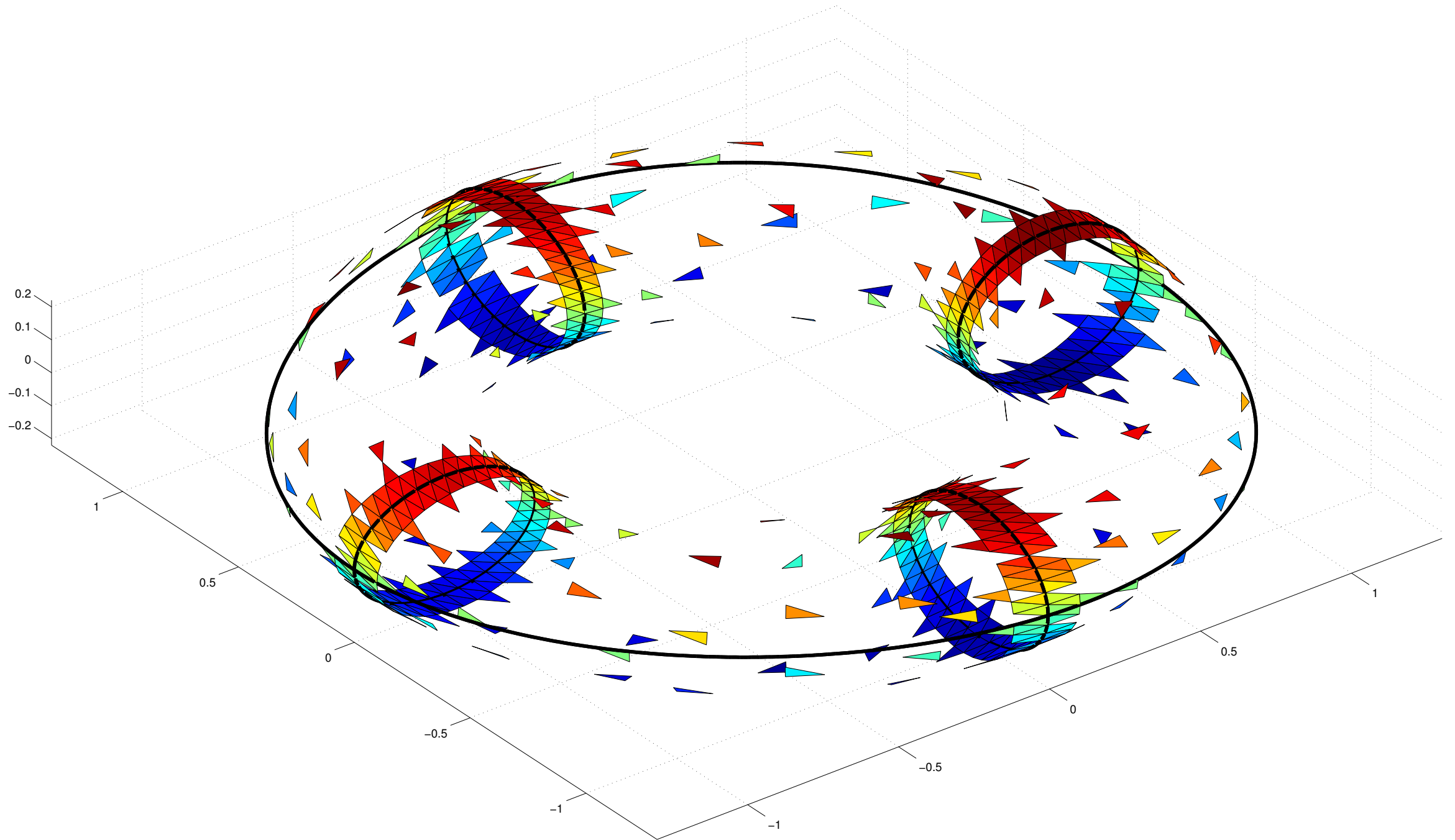
The reduced matrix represents a Nyström discretization supported on the panels shown.

The domain in physical space



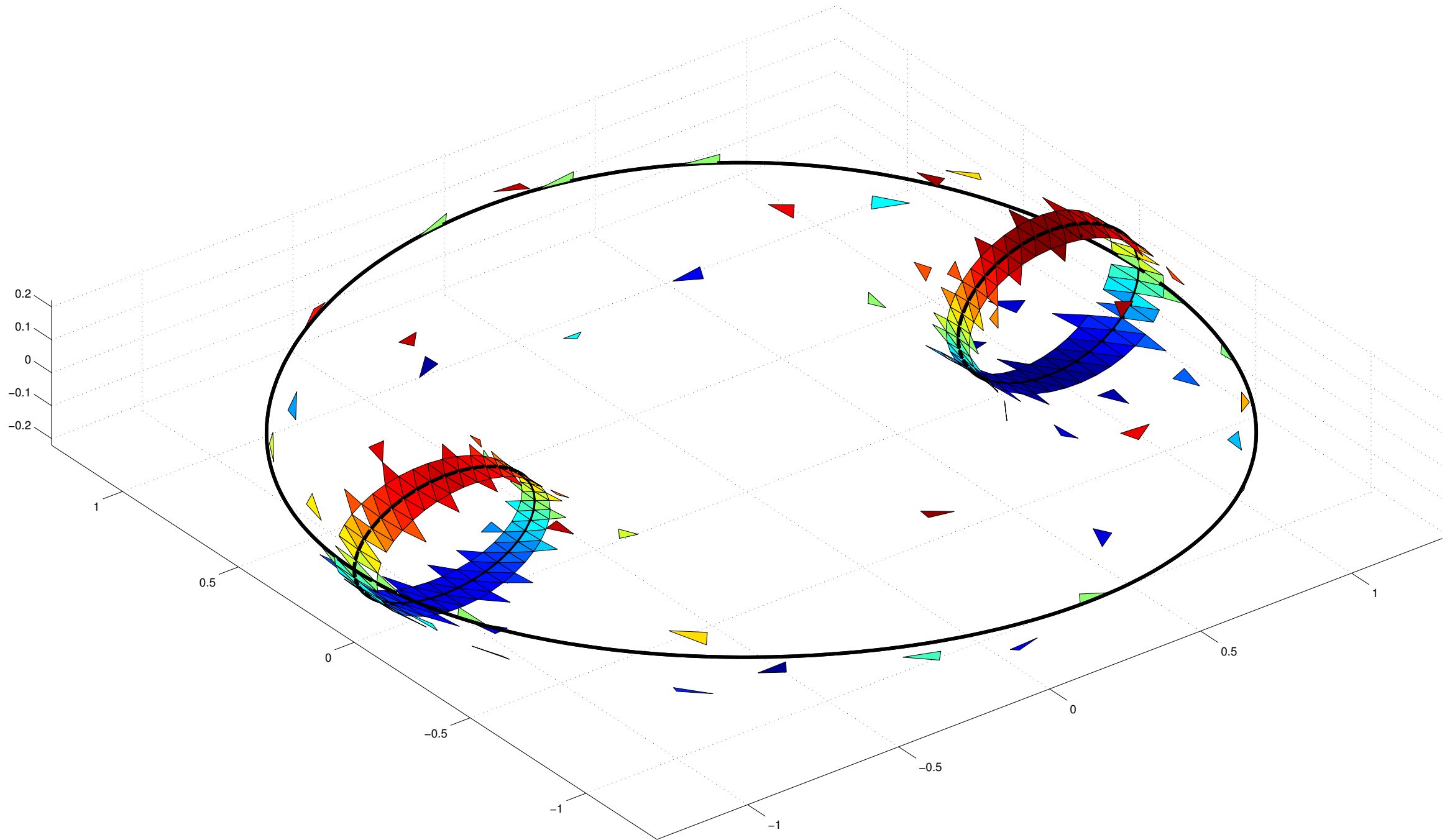
The reduced matrix represents a Nyström discretization supported on the panels shown.

The domain in physical space



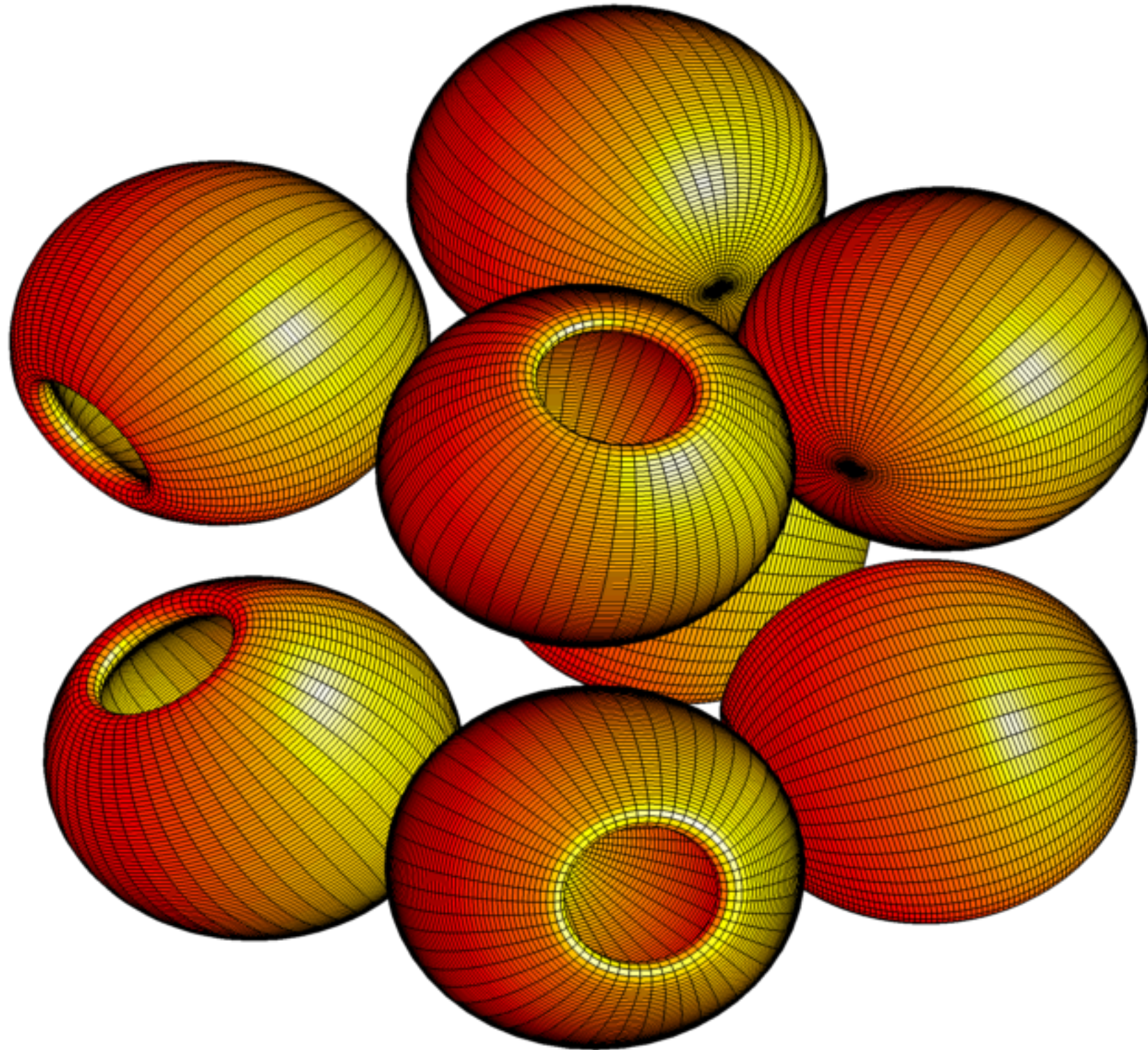
The reduced matrix represents a Nyström discretization supported on the panels shown.

The domain in physical space



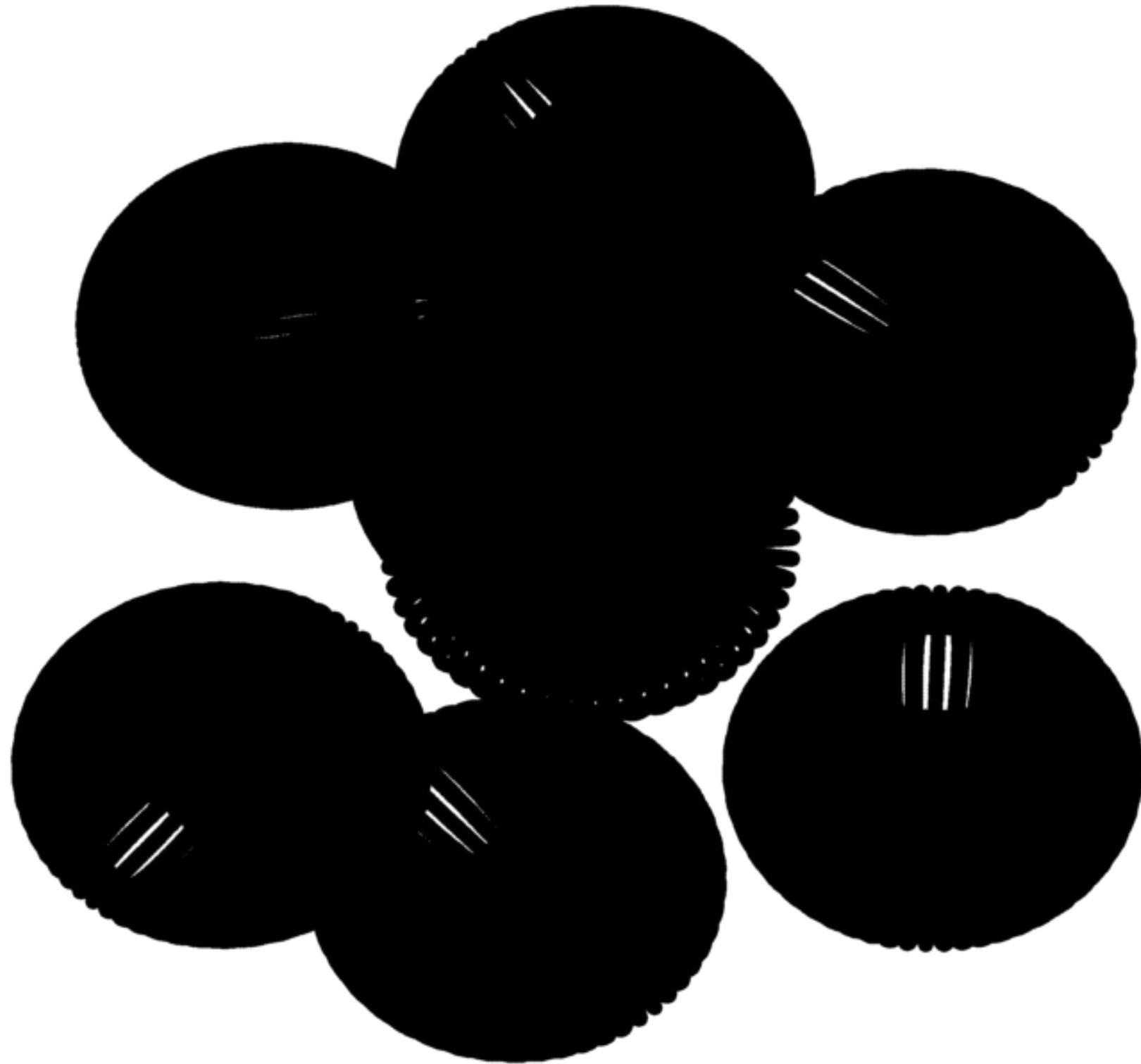
The reduced matrix represents a Nyström discretization supported on the panels shown.

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



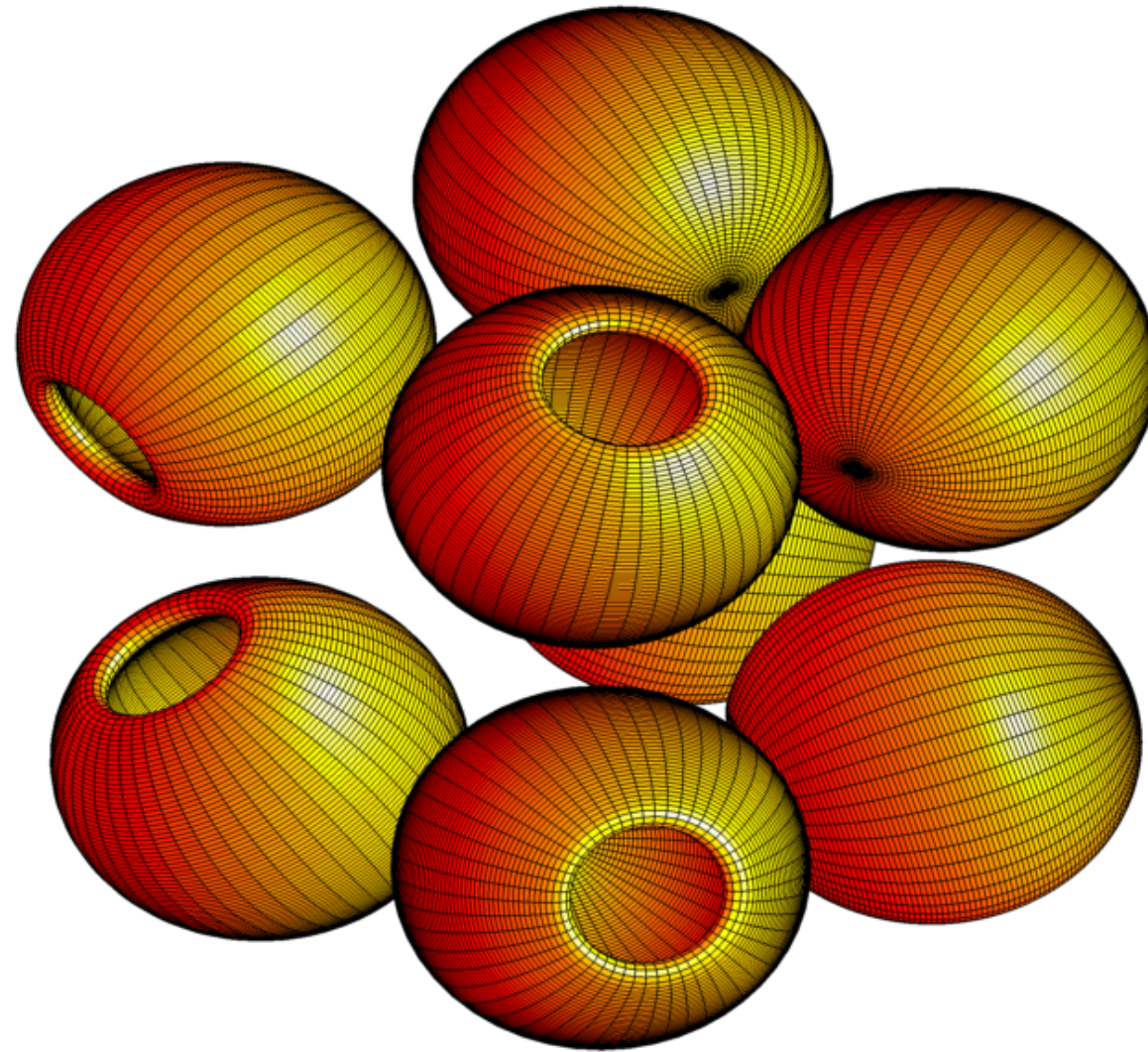
There are lots of discretization nodes involved. Very computationally intense!

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}	$T_{\text{precompute}}$	l_{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

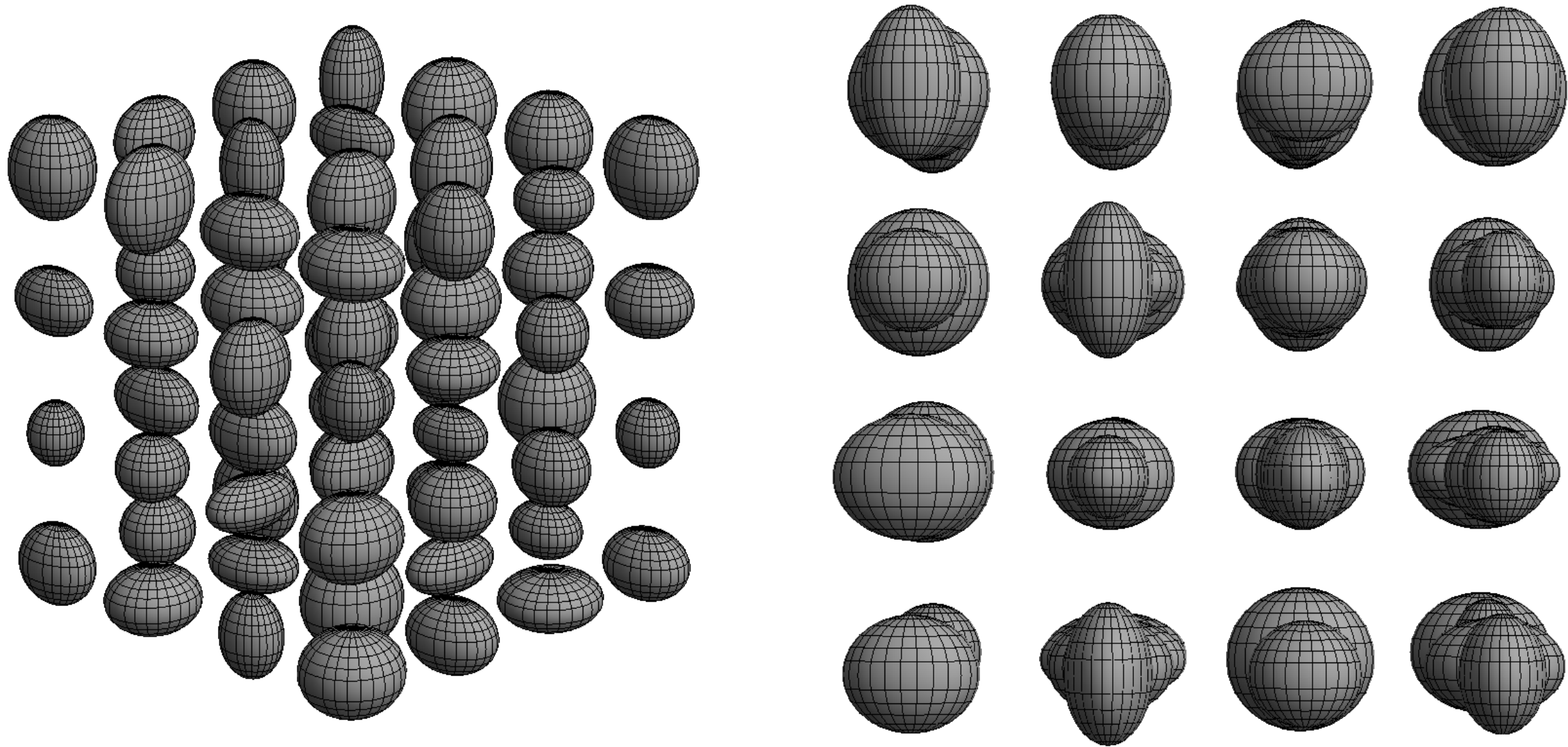
*Exterior **Helmholtz** problem solved on multibody bowl domain.*

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

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

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

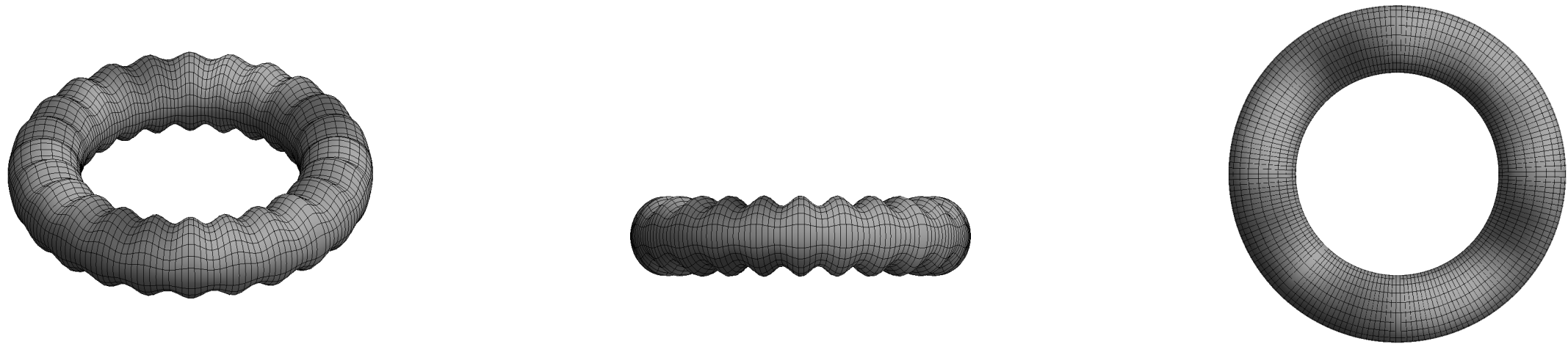
Grid dimensions	N	T	E	Ratio	Predicted
$2 \times 2 \times 2$	12 288	$1.02 \times 10^{+1}$	3.37×10^{-04}	-	-
$3 \times 3 \times 3$	41 472	$3.43 \times 10^{+1}$	4.81×10^{-04}	3.4	6.2
$4 \times 4 \times 4$	98 304	$7.92 \times 10^{+1}$	1.57×10^{-04}	2.3	3.7
$6 \times 6 \times 6$	331 776	$2.96 \times 10^{+2}$	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.)

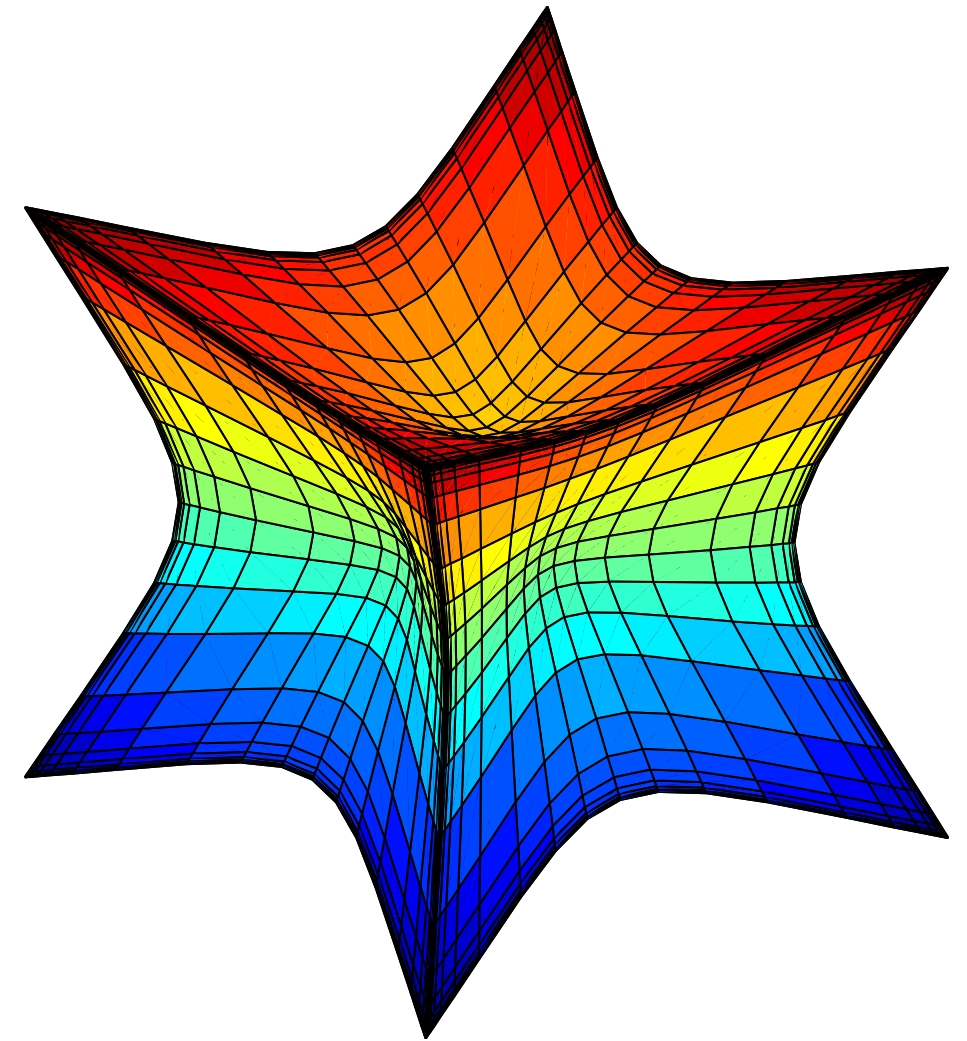
Numerical example — BIE on “edgy” surface (2013, with J. Bremer and A. Gillman)

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.

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:

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

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

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

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

Hierarchical Poincaré-Steklov Method: “FEM-BEM coupling”

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) v(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2 \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0, \end{cases}$$

where

- b is a smooth scattering potential with **compact support**, where
- v is a given “incoming potential” and where
- u is the sought “outgoing potential.”

$$-\Delta u - \kappa^2 (1-b)u = -\kappa^2 b v$$

support(b)

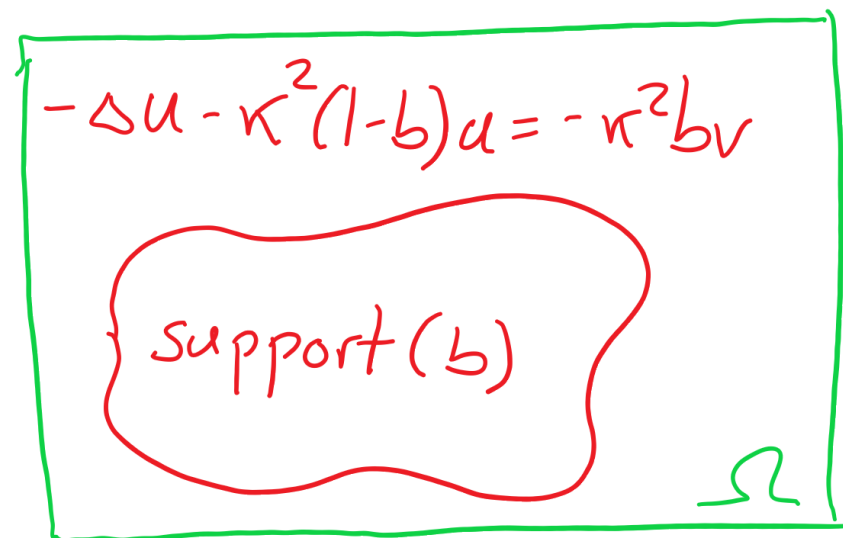
Hierarchical Poincaré-Steklov Method: “FEM-BEM coupling”

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) v(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2 \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0, \end{cases}$$

where

- b is a smooth scattering potential with **compact support**, where
- v is a given “incoming potential” and where
- u is the sought “outgoing potential.”



$$-\Delta u - \kappa^2 u = 0 \text{ on } \Omega^c$$

Introduce an artificial box Ω such that $\text{support}(b) \subseteq \Omega$.

Hierarchical Poincaré-Steklov Method: “FEM-BEM coupling”

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) v(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2 \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0, \end{cases}$$

where

- b is a smooth scattering potential with **compact support**, where
- v is a given “incoming potential” and where
- u is the sought “outgoing potential.”

$$-\Delta u - \kappa^2 (1 - b) u = -\kappa^2 b v$$

$$-\Delta u - \kappa^2 u = 0 \text{ on } \Omega^c$$

Introduce an artificial box Ω such that $\text{support}(b) \subseteq \Omega$.

On Ω :

- Variable coefficient PDE.

On Ω^c :

- Constant coefficient PDE.

Hierarchical Poincaré-Steklov Method: “FEM-BEM coupling”

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) v(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2 \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0, \end{cases}$$

where

- b is a smooth scattering potential with **compact support**, where
- v is a given “incoming potential” and where
- u is the sought “outgoing potential.”

$$-\Delta u - \kappa^2 (1 - b) u = -\kappa^2 b v$$

$$-\Delta u - \kappa^2 u = 0 \text{ on } \Omega^c$$

Introduce an artificial box Ω such that $\text{support}(b) \subseteq \Omega$.

On Ω :

- Variable coefficient PDE.
- Use HPS.

On Ω^c :

- Constant coefficient PDE.
- Use BIE.

Hierarchical Poincaré-Steklov Method: “FEM-BEM coupling”

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) v(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2 \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0, \end{cases}$$

where

- b is a smooth scattering potential with **compact support**, where
- v is a given “incoming potential” and where
- u is the sought “outgoing potential.”

$$-\Delta u - \kappa^2 (1 - b) u = -\kappa^2 b v$$

$$-\Delta u - \kappa^2 u = 0 \text{ on } \Omega^c$$

Introduce an artificial box Ω such that $\text{support}(b) \subseteq \Omega$.

On Ω :

- Variable coefficient PDE.
- Use HPS.
- Build DtN for $\partial\Omega$.

On Ω^c :

- Constant coefficient PDE.
- Use BIE.
- Build DtN for $\partial\Omega^c$.

Hierarchical Poincaré-Steklov Method: “FEM-BEM coupling”

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) v(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^2 \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u(\mathbf{x}) - i\kappa u(\mathbf{x})) = 0, \end{cases}$$

where

- b is a smooth scattering potential with **compact support**, where
- v is a given “incoming potential” and where
- u is the sought “outgoing potential.”

$$-\Delta u - \kappa^2 (1 - b) u = -\kappa^2 b v$$

$$-\Delta u - \kappa^2 u = 0 \text{ on } \Omega^c$$

Introduce an artificial box Ω such that $\text{support}(b) \subseteq \Omega$.

On Ω :

- Variable coefficient PDE.
- Use HPS.
- Build DtN for $\partial\Omega$.

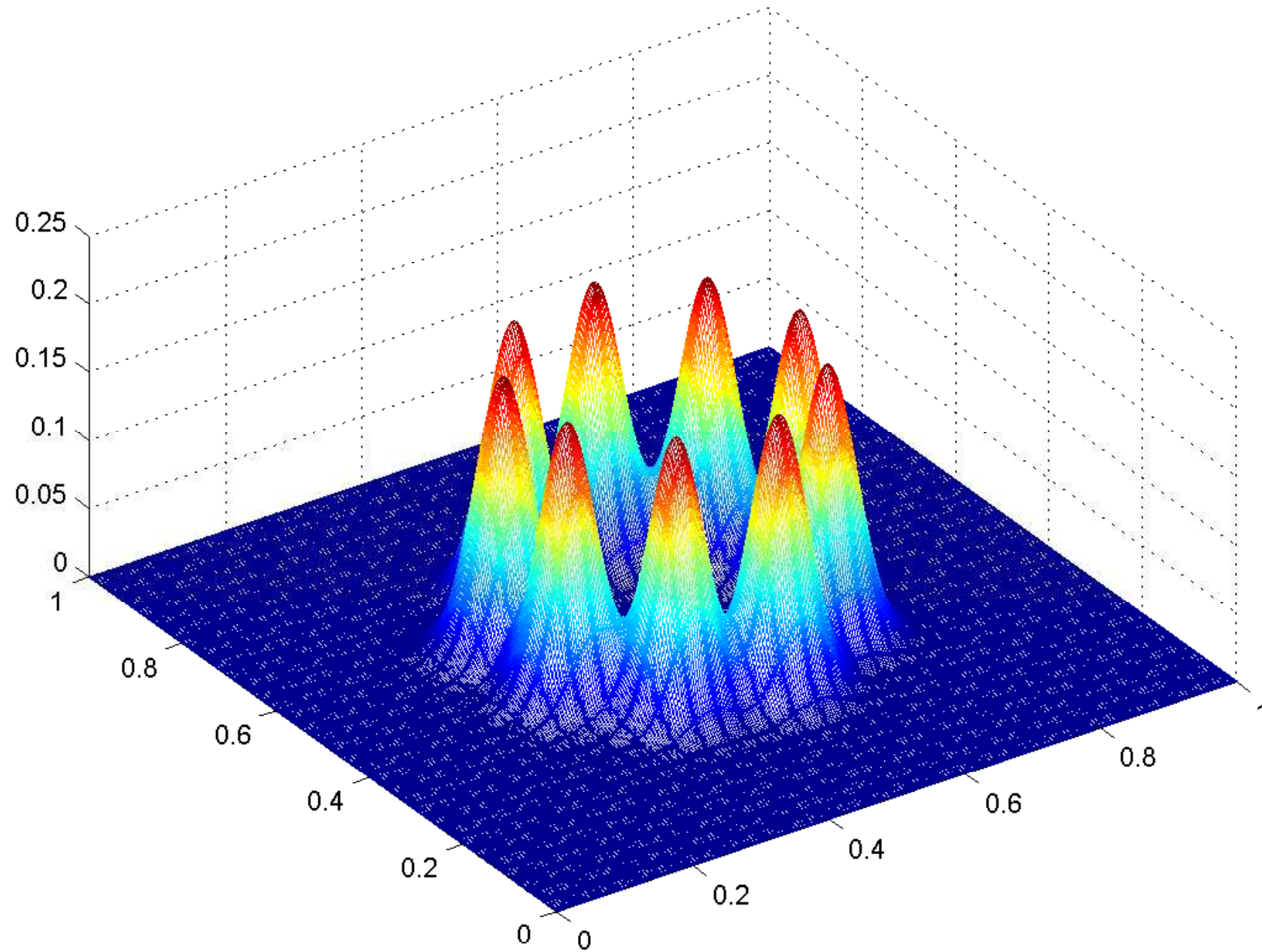
On Ω^c :

- Constant coefficient PDE.
- Use BIE.
- Build DtN for $\partial\Omega^c$.

• Merge using fast operator algebra!

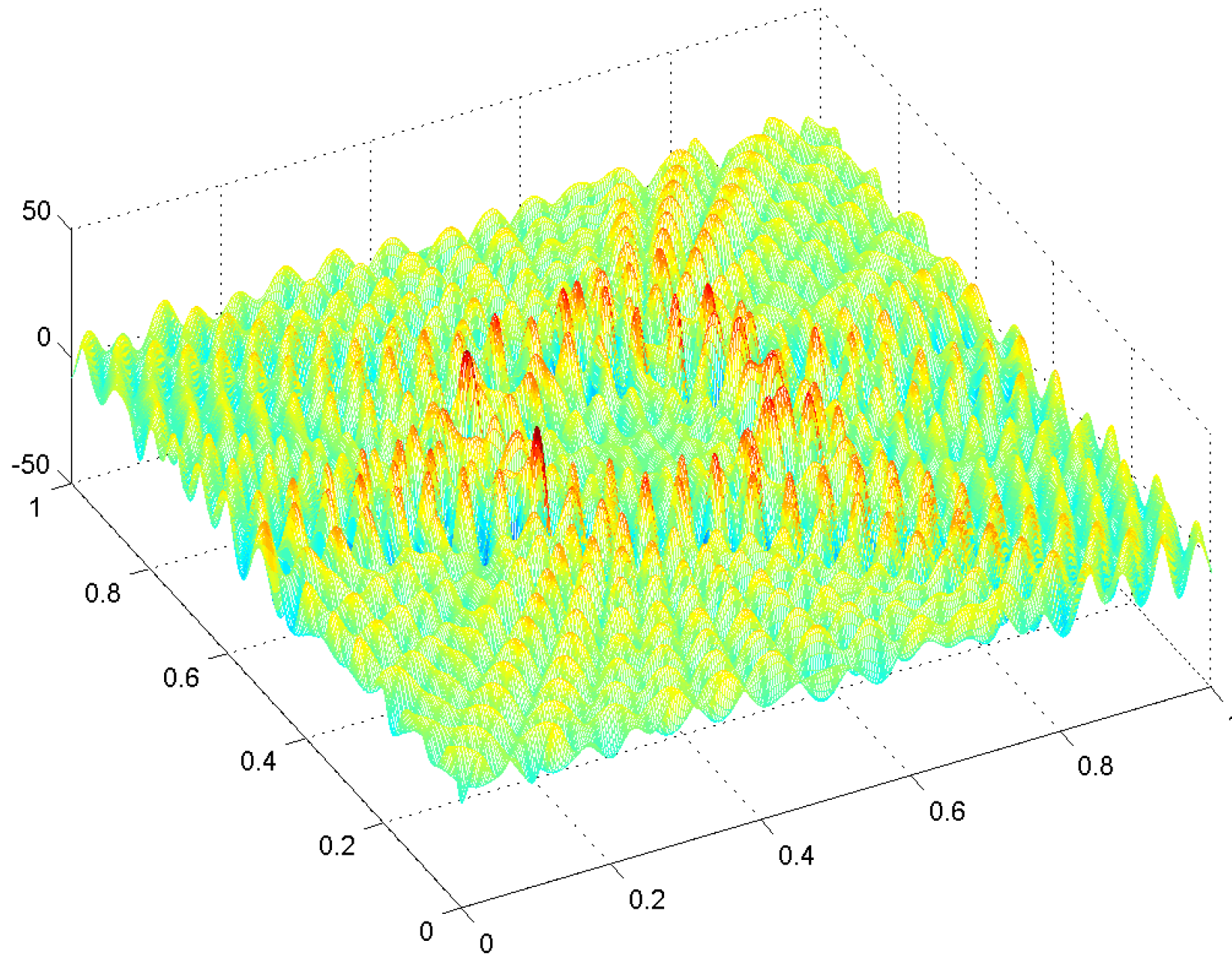
Example: Free space scattering
$$\begin{cases} -\Delta u_{\text{out}}(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u_{\text{out}}(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) u_{\text{in}}(\mathbf{x}) \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u_{\text{out}}(\mathbf{x}) - i\kappa u_{\text{out}}(\mathbf{x})) = 0 \end{cases}$$

The scattering potential b



Example: Free space scattering
$$\begin{cases} -\Delta u_{\text{out}}(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u_{\text{out}}(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) u_{\text{in}}(\mathbf{x}) \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u_{\text{out}}(\mathbf{x}) - i\kappa u_{\text{out}}(\mathbf{x})) = 0 \end{cases}$$

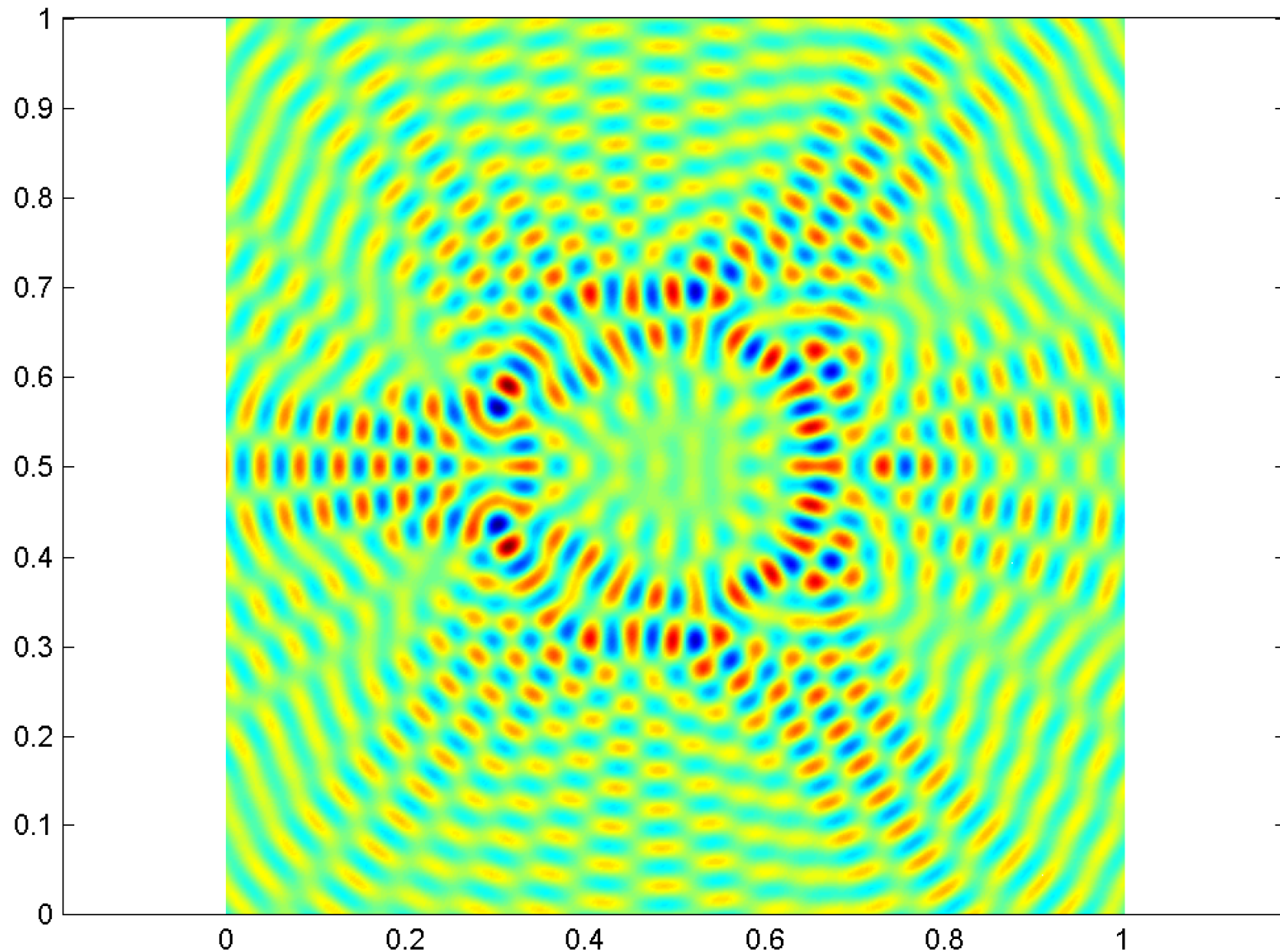
The outgoing field u_{out} (resulting from an incoming plane wave $u_{\text{in}}(\mathbf{x}) = \cos(\kappa x_1)$)



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

Example: Free space scattering
$$\begin{cases} -\Delta u_{\text{out}}(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u_{\text{out}}(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) u_{\text{in}}(\mathbf{x}) \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u_{\text{out}}(\mathbf{x}) - i\kappa u_{\text{out}}(\mathbf{x})) = 0 \end{cases}$$

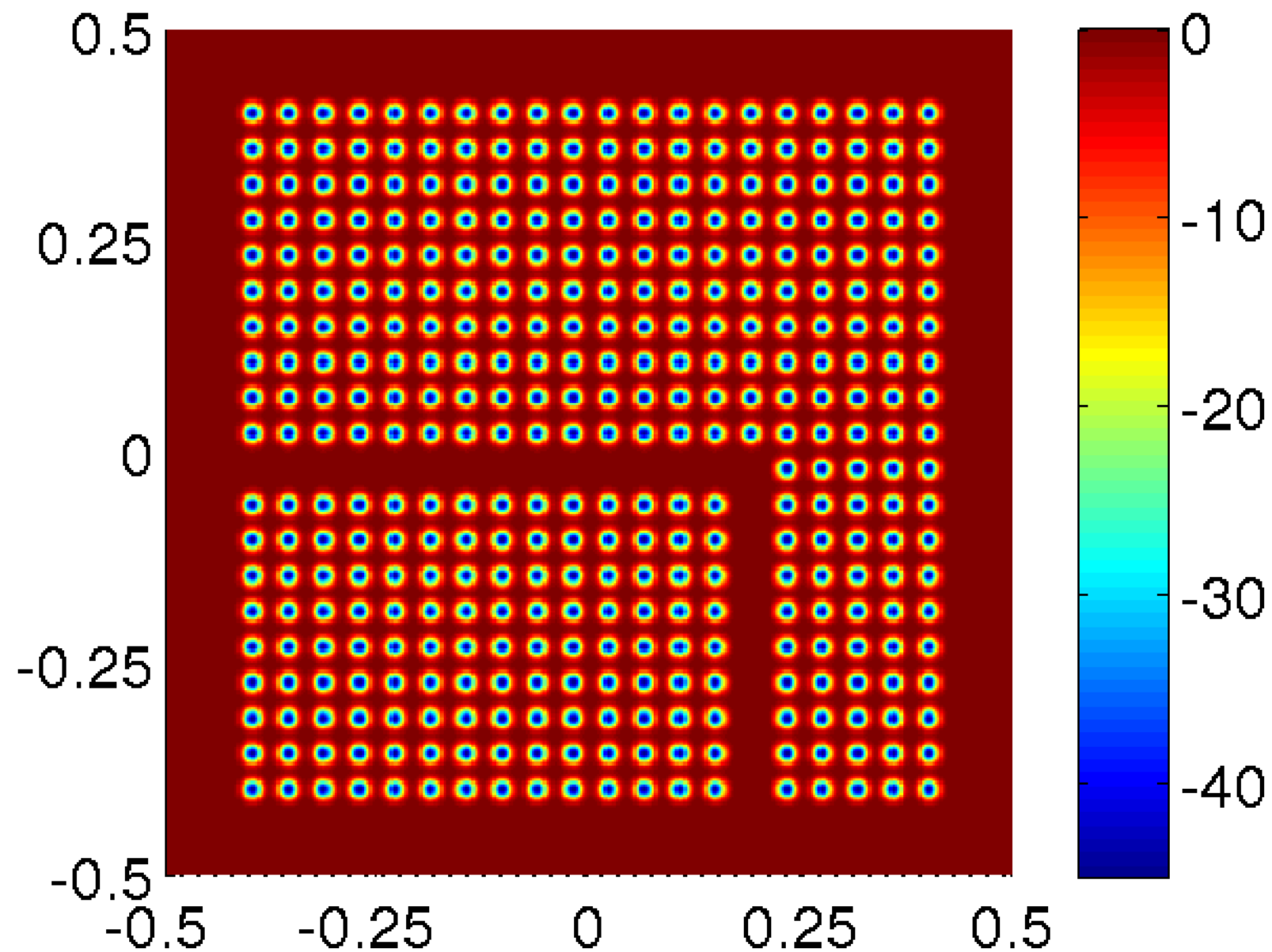
The outgoing field u_{out} (resulting from an incoming plane wave $u_{\text{in}}(x) = \cos(\kappa x_1)$)



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

Example: Free space scattering
$$\begin{cases} -\Delta u_{\text{out}}(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u_{\text{out}}(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) u_{\text{in}}(\mathbf{x}) \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u_{\text{out}}(\mathbf{x}) - i\kappa u_{\text{out}}(\mathbf{x})) = 0 \end{cases}$$

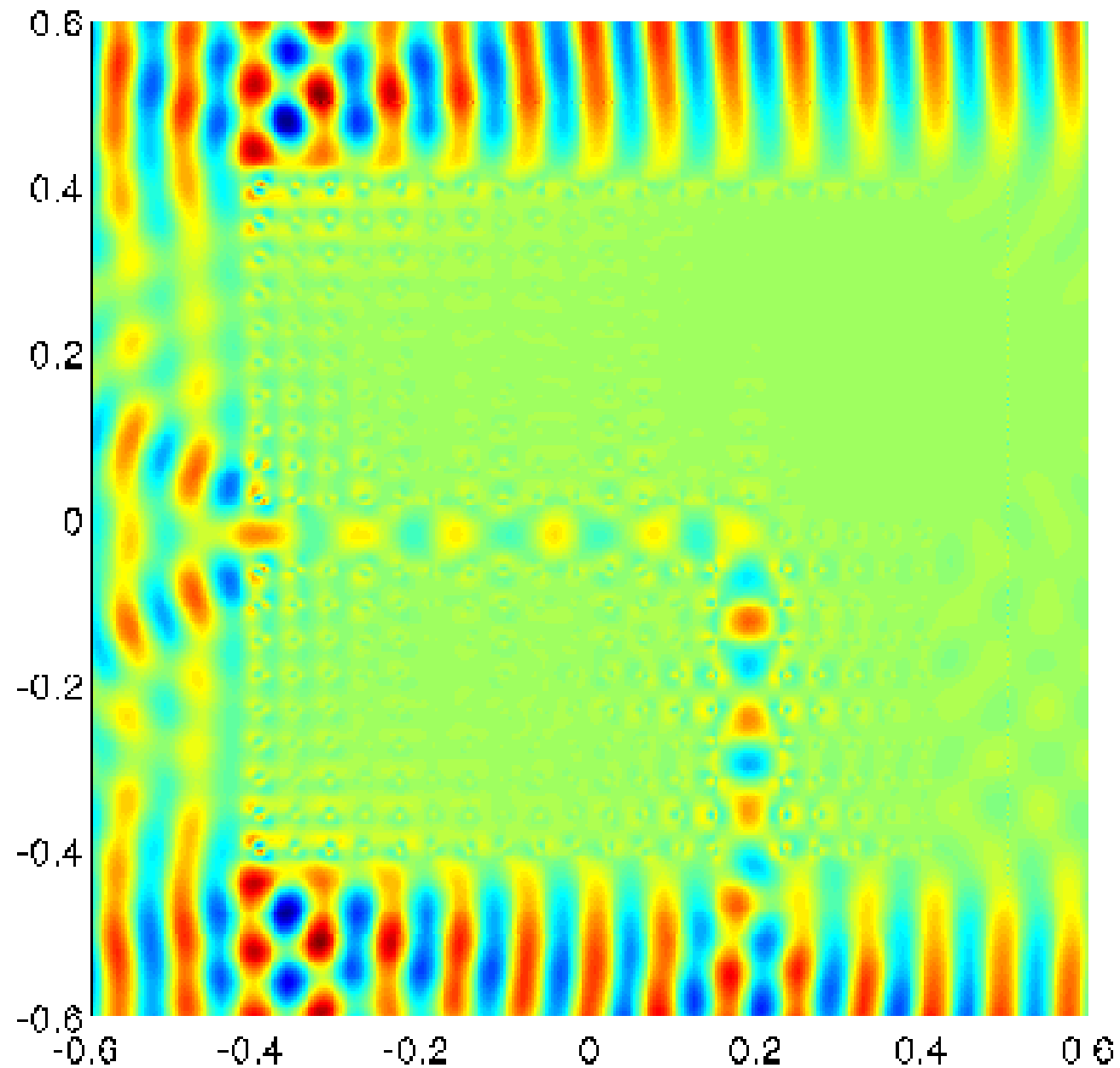
The scattering potential b — now a photonic crystal with a wave guide.



$N = 231\,361$ $T_{\text{build}} = 7.2 \text{ sec}$ $T_{\text{solve}} = 0.06 \text{ sec}$ $E \approx 10^{-6}$ (estimated)

Example: Free space scattering
$$\begin{cases} -\Delta u_{\text{out}}(\mathbf{x}) - \kappa^2 (1 - b(\mathbf{x})) u_{\text{out}}(\mathbf{x}) = -\kappa^2 b(\mathbf{x}) u_{\text{in}}(\mathbf{x}) \\ \lim_{|\mathbf{x}| \rightarrow \infty} \sqrt{|\mathbf{x}|} (\partial_{|\mathbf{x}|} u_{\text{out}}(\mathbf{x}) - i\kappa u_{\text{out}}(\mathbf{x})) = 0 \end{cases}$$

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



Randomized SVD (RSVD)

Joint work with V. Rokhlin and M. Tygert (2005)

The type of direct solver described spends a lot of the execution time on computing approximate low rank factorizations to matrices. Let us describe how such computations can be greatly accelerated using **randomized** methods.

Model problem: Let \mathbf{A} be a given $m \times n$ matrix, and let k be a target rank such that $k \ll \min(m, n)$. Then suppose that we seek to compute an approximate partial SVD:

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

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

Randomized SVD (RSVD)

Joint work with V. Rokhlin and M. Tygert (2005)

The type of direct solver described spends a lot of the execution time on computing approximate low rank factorizations to matrices. Let us describe how such computations can be greatly accelerated using **randomized** methods.

Model problem: Let \mathbf{A} be a given $m \times n$ matrix, and let k be a target rank such that $k \ll \min(m, n)$. Then suppose that we seek to compute an approximate partial SVD:

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

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

Input: An $m \times n$ matrix \mathbf{A} , a target rank k , and an over-sampling parameter p (say $p = 5$).

Output: Rank- $(k + p)$ factors \mathbf{U} , \mathbf{D} , and \mathbf{V} in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.

(1) Draw an $n \times (k + p)$ **random matrix** \mathbf{R} .

(2) Form the $m \times (k + p)$ **sample matrix** $\mathbf{Y} = \mathbf{AR}$.

(3) Compute an **ON matrix** \mathbf{Q} s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Input: An $m \times n$ matrix \mathbf{A} , a target rank k , and an over-sampling parameter p (say $p = 5$).

Output: Rank- $(k + p)$ factors \mathbf{U} , \mathbf{D} , and \mathbf{V} in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.

(1) Draw an $n \times (k + p)$ **random matrix** \mathbf{R} .

(2) Form the $m \times (k + p)$ **sample matrix** $\mathbf{Y} = \mathbf{AR}$.

(3) Compute an **ON matrix** \mathbf{Q} s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(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.
- Analogous schemes exist for computing a partial QR factorization, or a so called “interpolative decomposition” where a number of the columns/rows are chosen to serve as a basis for the column/row space.
- 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.
For instance, use the sampling matrix $\mathbf{Y} = \mathbf{AA}^*\mathbf{AG}$ instead of $\mathbf{Y} = \mathbf{AG}$.
- We can reduce the flop count from $O(mnk)$ to $O(mn \log k)$ by using a so called “fast Johnson-Lindenstrauss” transform. Practical speed gain too!

Input: An $m \times n$ matrix \mathbf{A} , a target rank k , and an over-sampling parameter p (say $p = 5$).

Output: Rank- $(k + p)$ factors \mathbf{U} , \mathbf{D} , and \mathbf{V} in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.

(1) Draw an $n \times (k + p)$ **random matrix** \mathbf{R} .

(2) Form the $m \times (k + p)$ **sample matrix** $\mathbf{Y} = \mathbf{AR}$.

(3) Compute an **ON matrix** \mathbf{Q} s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

The output of RSVD is a random variable, as it depends on the draw of \mathbf{R} . We have rigorous mathematical results describing the errors of the algorithm in expectation, as well as the risk of large deviations. Connections to random matrix theory.

Input: An $m \times n$ matrix \mathbf{A} , a target rank k , and an over-sampling parameter p (say $p = 5$).

Output: Rank- $(k + p)$ factors \mathbf{U} , \mathbf{D} , and \mathbf{V} in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.

(1) Draw an $n \times (k + p)$ **random matrix** \mathbf{R} .

(2) Form the $m \times (k + p)$ **sample matrix** $\mathbf{Y} = \mathbf{AR}$.

(3) Compute an **ON matrix** \mathbf{Q} s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\hat{\mathbf{V}}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

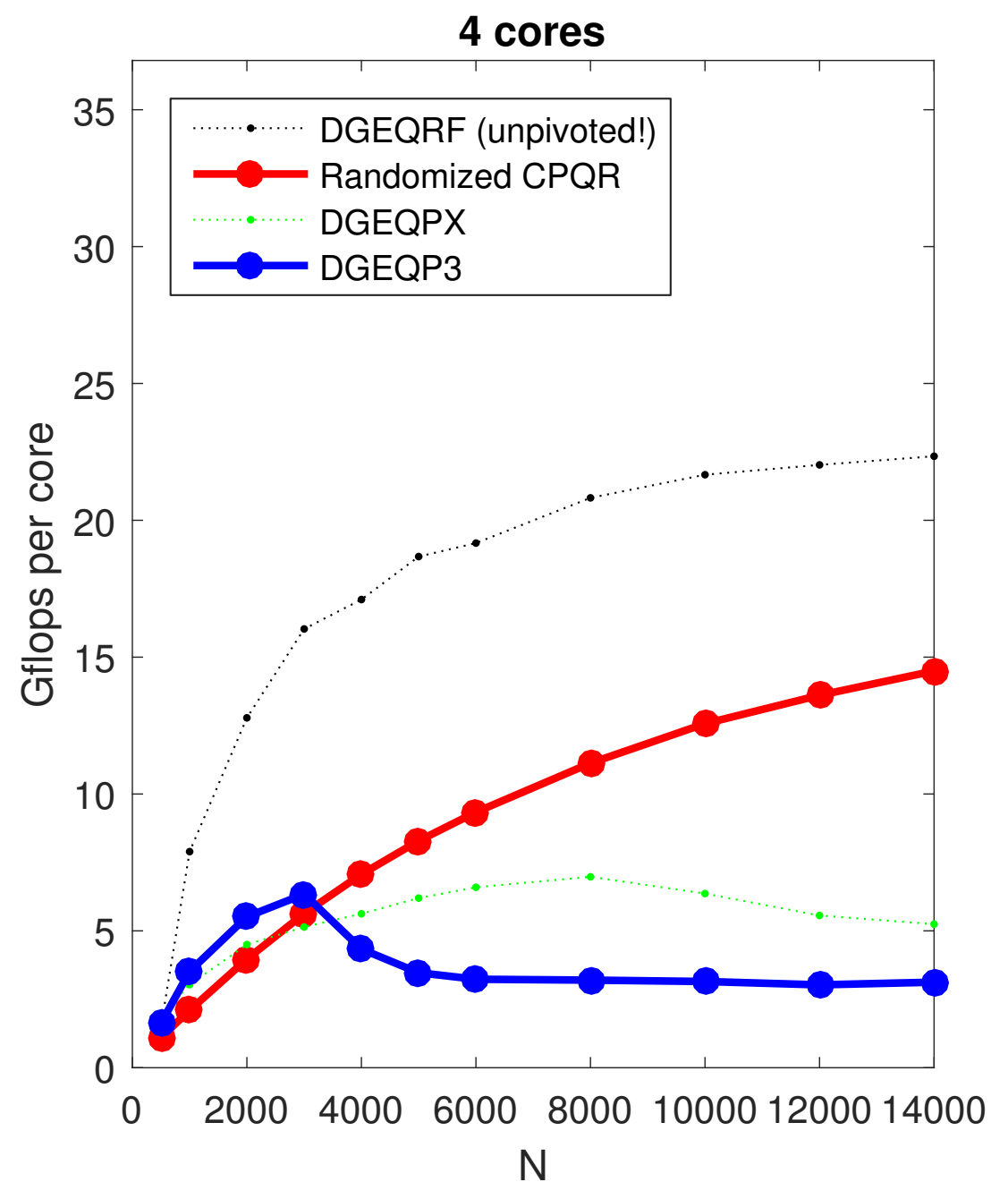
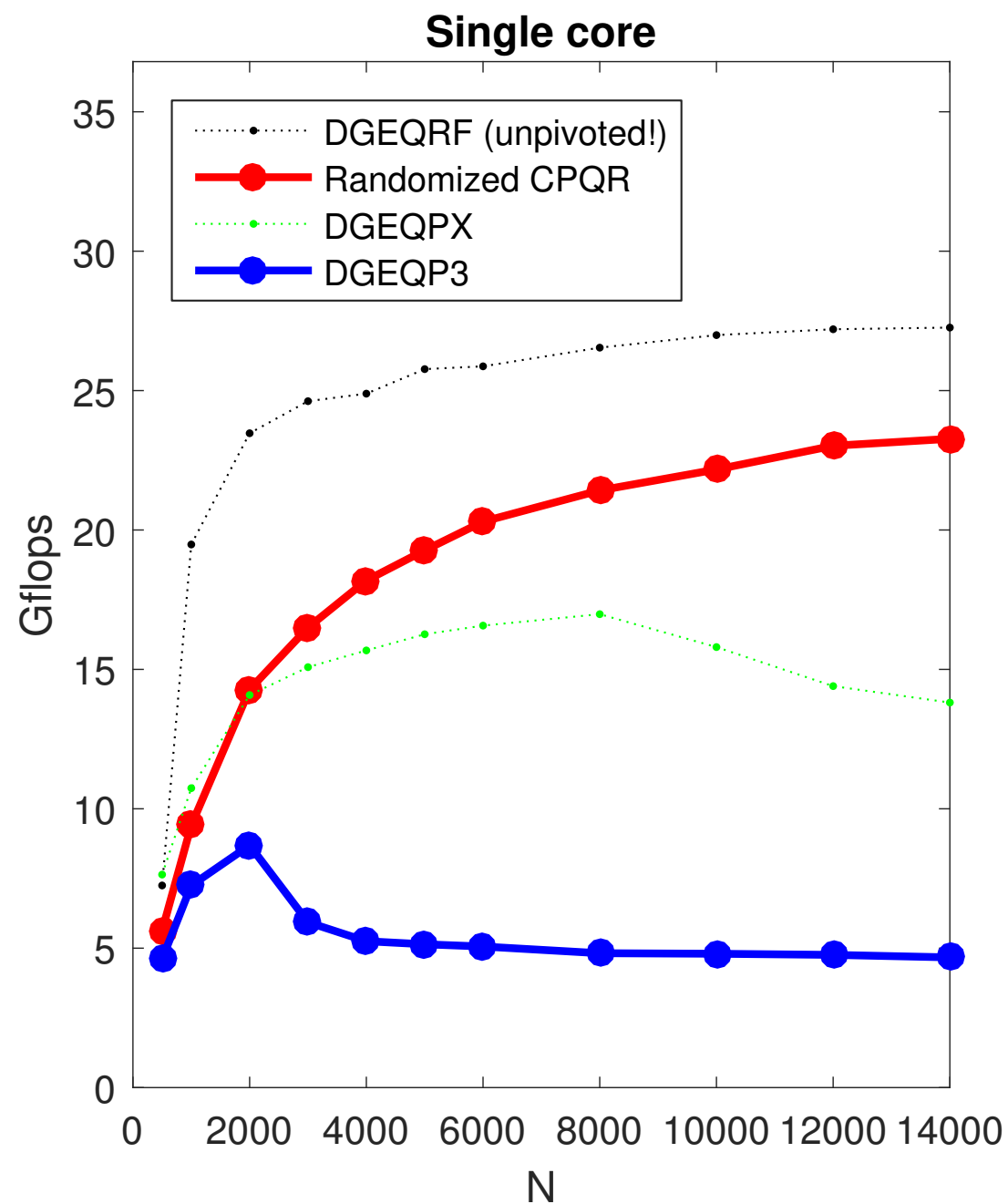
The output of RSVD is a random variable, as it depends on the draw of \mathbf{R} . We have rigorous mathematical results describing the errors of the algorithm in expectation, as well as the risk of large deviations. Connections to random matrix theory.

The perhaps most important feature of randomized algorithms is that they are very communication efficient. This makes them particularly competitive in strongly communication constrained environments (huge matrices stored out-of-core, distributed memory parallel computers, GPUs).

There exist **single-pass** versions of the RSVD that work even under the constraint that each matrix element can be viewed only once. (“Streaming algorithms.”)

Very recent result: Randomization can be used to greatly accelerate **full** rank-revealing factorizations such as the column pivoted QR factorization, or the UTV factorization.

The gain is attained due to decreased communication, not fewer flops.



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

We use the term *rank-structured* to describe a matrix whose off-diagonal blocks have low rank to some given precision. There are many different “flavors,” including:

- \mathcal{H} - and \mathcal{H}^2 -matrices of Hackbusch and co-workers. This work represents the first systematic attack.
- Generalizations of the Fast Multipole Method (FMM): kernel-independent FMMs, inverse FMM, ASKIT, etc.
- Hierarchically Block Separable (HBS) matrices, a.k.a. “HSS” matrices.
- HODLR matrices (a.k.a. \mathcal{S} -matrices).

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

Objective: Suppose a matrix \mathbf{A} is rank-structured, that you are *given* a tessellation pattern, and that you have an efficient technique for evaluating the matrix-vector product $\mathbf{x} \mapsto \mathbf{Ax}$. We then seek to build all factors in the rank-structured representation of \mathbf{A} .

Applications: Build “frontal matrices” in nested dissection. Matrix-matrix multiplication of two structured matrices. Convert from, say, FMM format, to HBS format. Et cetera.

Let \mathbf{A} be a rank-structured matrix, for which we can rapidly evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$.

Case 1: Suppose that in addition to matvec, we can also evaluate individual entries of \mathbf{A} . Then an HBS (a.k.a. HSS) representation can be computed in $O(N)$ operations.

Very computationally efficient — requires only one call each to \mathbf{A} and \mathbf{A}^* matvecs to two sets of, say, $k + 10$ vectors.

- 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. Distributed memory implementations exist, etc.

Case 2: If all we have is the matvec, then we can still compute a rank-structured representation of \mathbf{A} using a so called “peeling” algorithms. The price we have to pay is that we now need $O(k \times \log N)$ matvecs involving \mathbf{A} and \mathbf{A}^* .

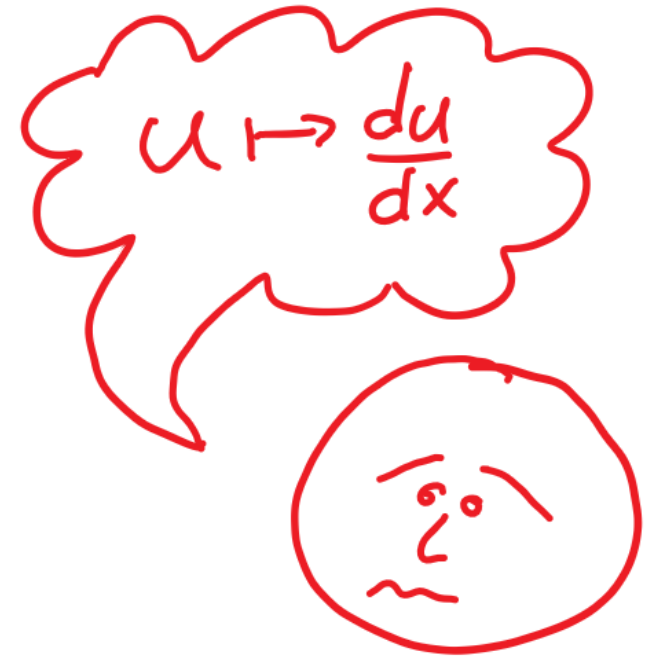
The method is still very fast in many situations, and can save messy coding work. For instance, implementing the matrix-matrix multiplication, or changing the partition tree, are quite hard to implement efficiently.

- 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.” SISC 2016.

Key Ideas:

The solution operator of a linear elliptic PDE is “friendly.”

- Smoothing.
- Stable.



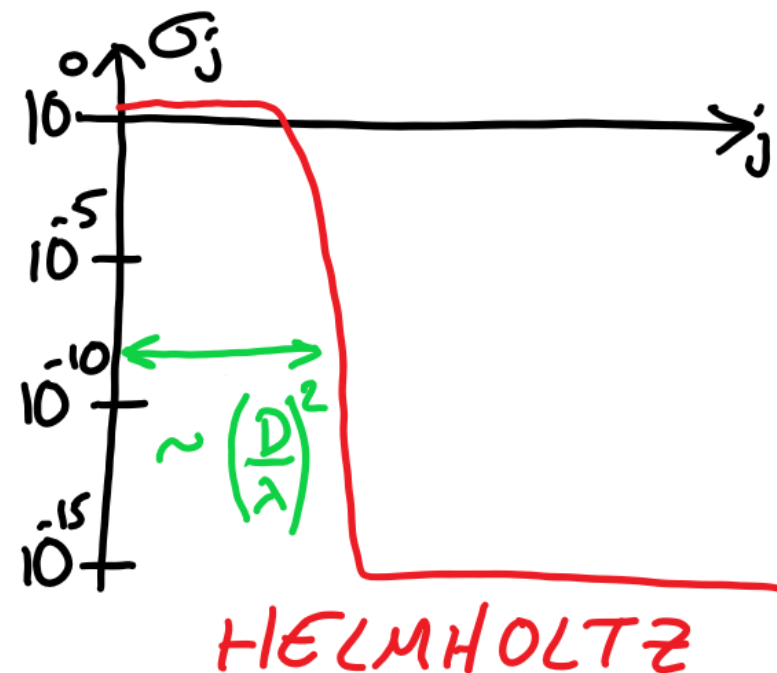
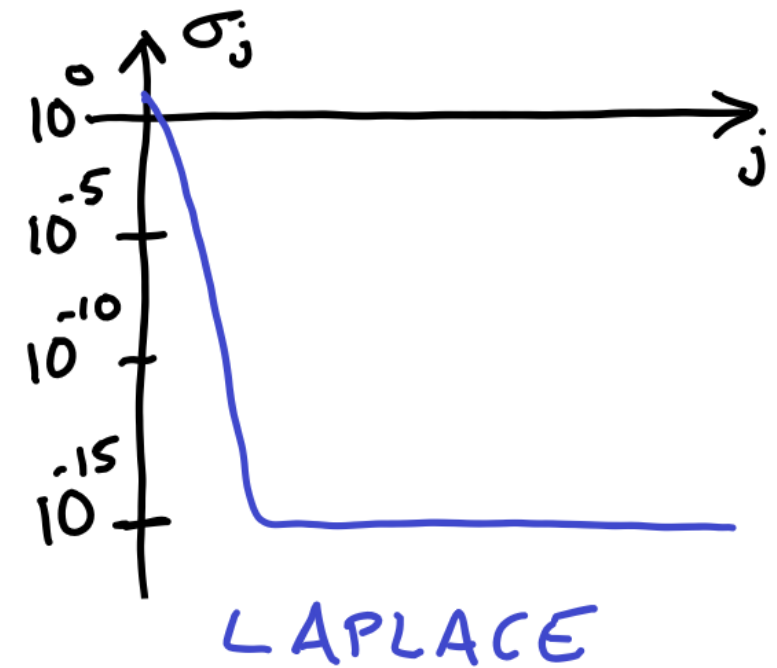
Key Ideas:

The solution operator of a linear elliptic PDE is “friendly.”

- Smoothing.
- Stable.

Long range interactions are low rank.

- Cf. St Venant principle, multipole expansions, etc.
- Smoothness is *not* necessary.
- Numerical compression is essential.
- Wave problems with small λ remain challenging.



Key Ideas:

The solution operator of a linear elliptic PDE is “friendly.”

- Smoothing.
- Stable.

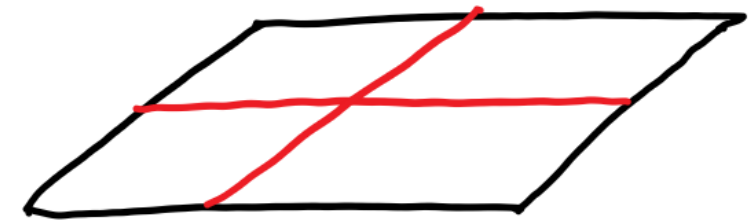
Long range interactions are low rank.

- Cf. St Venant principle, multipole expansions, etc.
- Smoothness is *not* necessary.
- Numerical compression is essential.
- Wave problems with small λ remain challenging.

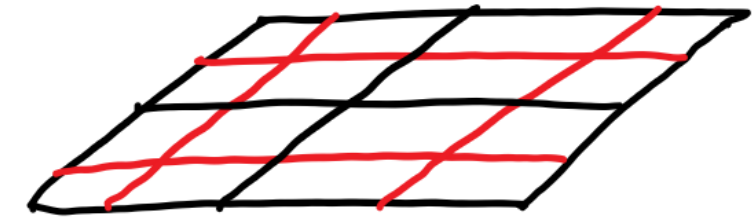
Hierarchical Divide-and-Conquer.

- Generalizations of nested dissection.
- Need *double* hierarchies for $O(N)$ complexity.
- Formulations that are inherently well-conditioned exist.

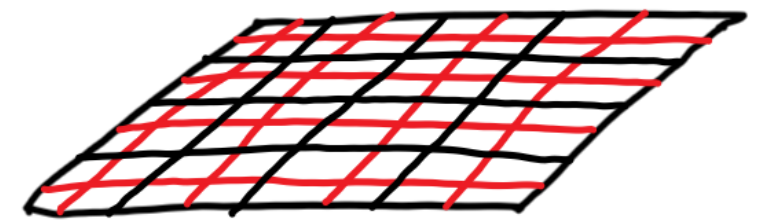
New randomized methods for matrix algebra → acceleration & simplification.



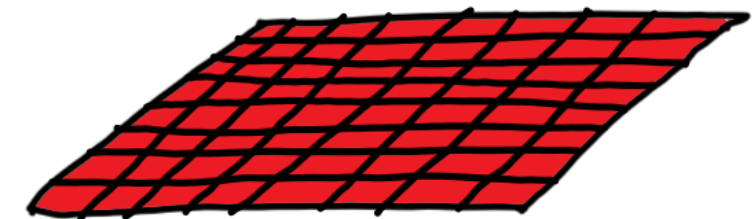
Level 0



Level 1



Level 2



Level 3

Where we are now:

- We have developed direct solvers with $O(N)$ complexity 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. (Work in progress ...)
- Advantages of direct solvers:
 - Often instantaneous solves once a solution operator has been built.
 - Can eliminate problems with slow convergence of iterative solvers.
 - Communication efficient.
- Disadvantages of direct solvers:
 - Memory hogs. (But distributed memory is OK.)
 - The build stage is still slow for many 3D problems. (I am optimistic that we will fix this!)

Where to go next:

New powerful tool available → lots of opportunities!

- Explore happy couplings:
 - Direct solver + high order discretization. *(Helps with memory. Wave problems.)*
 - Direct solver + integral equation formulations. *(Need dense matrices anyway.)*
 - Direct solver + parallelization. *(Root of tree is cheap!)*
 - Direct solver + numerical coarse graining. *(Another talk...)*
- Parabolic and hyperbolic problems. Parallel-in-time methods?

For more details, a recent book has it all:

