Fast algorithms for global operators

Per-Gunnar Martinsson

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

University of Texas at Austin

Students, postdocs, collaborators: Daniel Appelö, Yunhui Cai, Chao Chen, Ke Chen, Yijun Dong, Adrianna Gillman, Abinand Gopal, James Levitt, Michael O'Neil, Heather Wilber, Bowei Wu, Anna Yesypenko.

Research support by:

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

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

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with 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:

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

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

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

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with 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:

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

Standard numerical recipe for (BVP): (1) Discretize via FD/FEM. (2) Iterative solver.

Focal point of this talk: The solution operator for (BVP). $\rightarrow$ Direct solvers.

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

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

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

Linear solution operators: As a warmup, let us consider the Poisson equation

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

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

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

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

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

In principle very simple. Numerically non-trivial, however: The operator is *global*, so discretizing it leads to a *dense* matrix. (There is also the singular kernel to worry about!)

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

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

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

Linear solution operators: A general solution operator for (BVP) takes the form

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

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

Good: The operators in (SLN) are 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.*

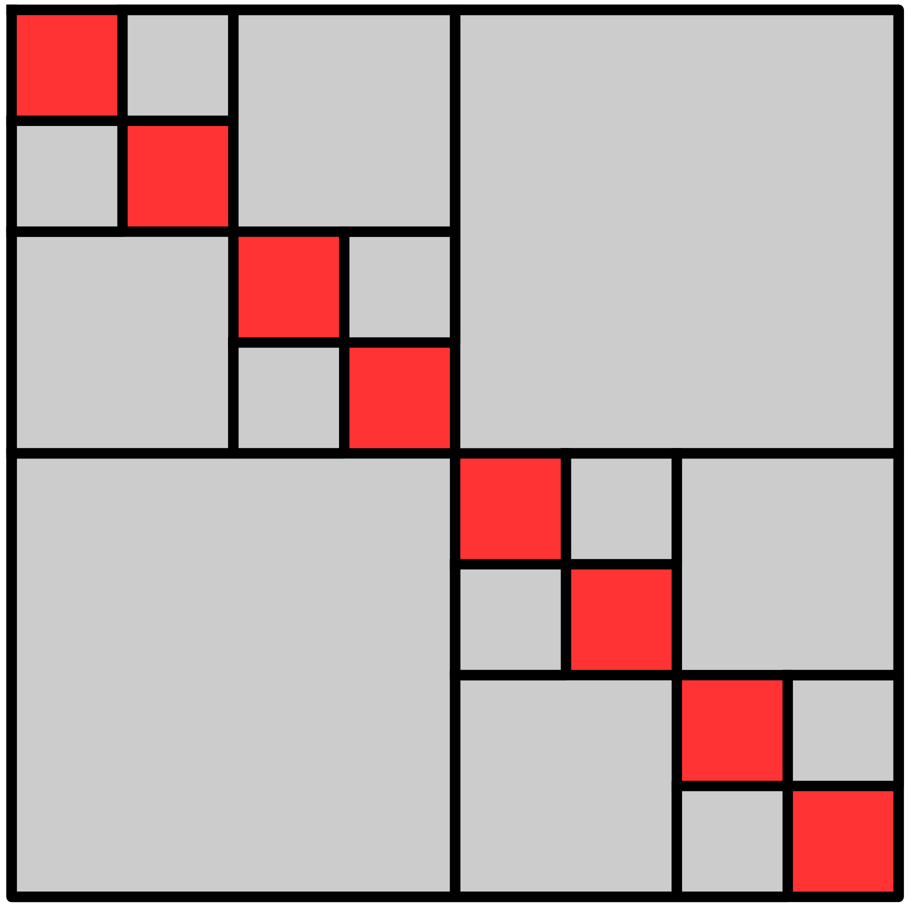
Dense matrices upon discretization. $O(N^2)$ cost? $O(N^3)$ cost?

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

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

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 $\mathcal{H}^2$-matrices (Hackbusch et al); Hierarchically Block Separable matrices; Hierarchically Semi Separable matrices (Xia et al); HODLR matrices (Darve et al); BLR matrices (Buttari, Amestoy, Mary, …); …

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

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

In real life, tessellation patterns of rank structured matrices tend to be more complex …

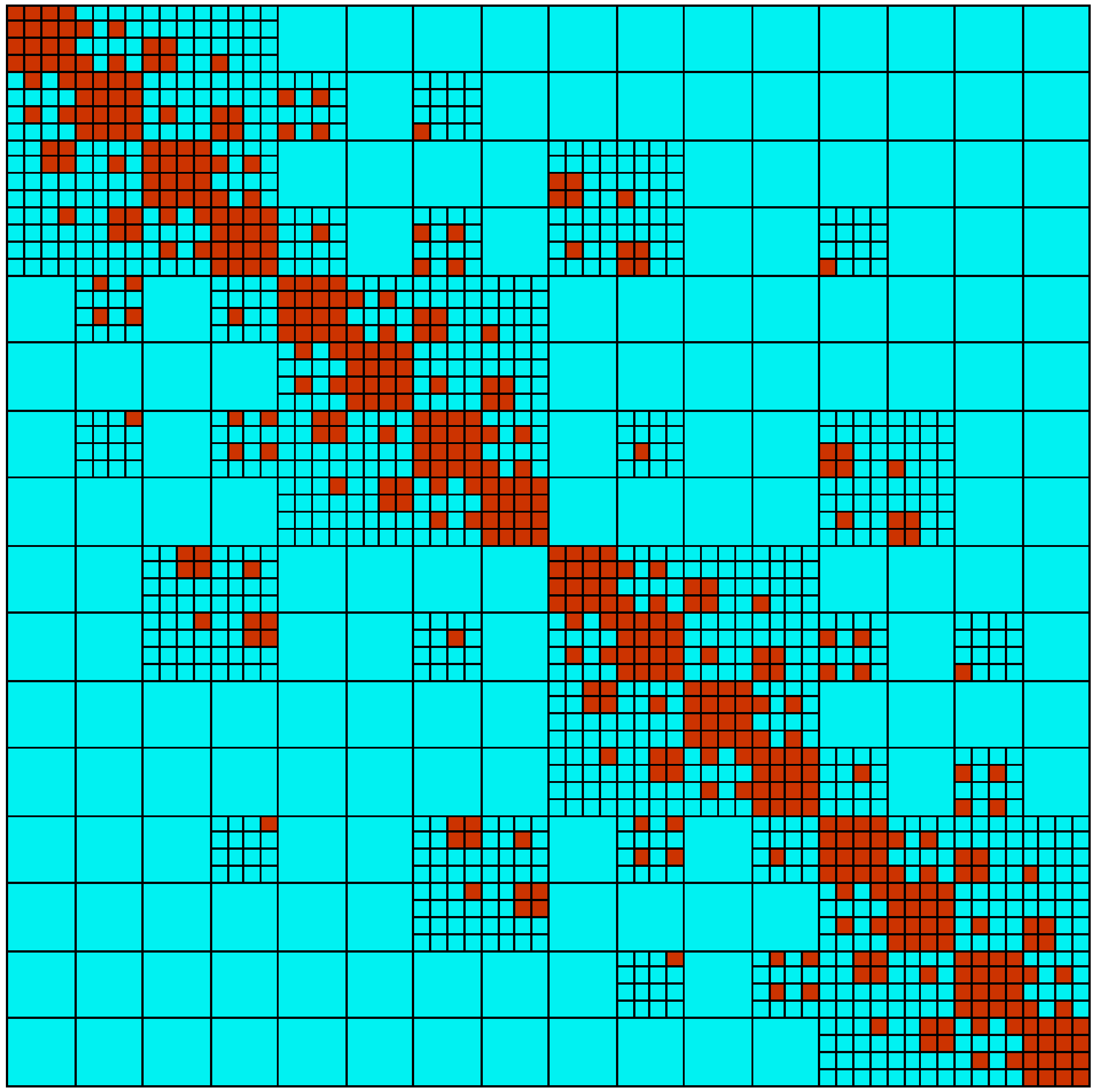


Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

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

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

The talk will describe recent work on algorithms that numerically construct an approximation to (SLN).

When using these algorithms, the process of solving (BVP) splits into two stages:

1. ***"Factorization" or "build" stage:*** Build a representation of the inverse operator.
2. ***"Solve" stage:*** Apply the computed inverse to given data f or (and) g.

Typical characteristics of methods of this type:

- Memory usage tends to be high.
- Stage 1 tends to be slower than an iterative solve, *when convergence is fast.*
- Stage 2 is almost always VERY fast.

Fast Direct Solvers (FDS) are most competitive when:

- Getting iterative methods to converge rapidly is hard.
- When the cost of Stage 1 can be amortized over many solves.

$\rightarrow$ scattering problems, time-stepping, optimization, ...

History:

1980s: Rokhlin and Greengard develop the Fast Multipole Method.

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

1996: Michielssen, Boag, Chew: Fast direct solvers for 3D scattering problems in certain geometries.

1998 onwards: Hackbusch, Bebendorf, Börm, Grasedyck, Khoromskij, Sauter, Tyrtyshnikov, …develop $\mathcal{H}$ and $\mathcal{H}^2$-frameworks that provide explicit recipes for operator algebra in $O(n \log^r n)$ operations for r moderate.

Outline of talk:

- Introduction: Problem formulation & solution operators. *[Done!]*
- Curse of dimensionality.
- Interaction ranks — why are they small? How small are they?
- (Versions of fast direct solvers — “strong” versus “weak” etc.)
- High order discretizations and fast direct solvers.
- Randomized low rank approximation.
- Randomized compression of rank structured matrices.

Curse of dimensionality

Algorithms involving rank-structured matrices scale *very* poorly with dimension.

For instance, for the classical Fast Multipole Method, key quantities scale as:

Dimension	Typical ranks	Number of "neighbors"	Length of "interaction list"
1	2	2	3
2	10–50	8	27
3	50–500	26	189
d	$(\log(1/\varepsilon))^{d-1}$	$3^d - 1$	$6^d - 3^d$

Curse of dimensionality

Algorithms involving rank-structured matrices scale *very* poorly with dimension.

For instance, for the classical Fast Multipole Method, key quantities scale as:

Dimension	Typical ranks	Number of "neighbors"	Length of "interaction list"
1	2	2	3
2	10–50	8	27
3	50–500	26	189
d	$(\log(1/\varepsilon))^{d-1}$	$3^d - 1$	$6^d - 3^d$

For fast direct solvers, the scaling with dimension is equally problematic:

Dimension	Current state of affairs
1	Extremely fast. Linear scaling is easy to attain.
2	Quite fast in practice, but simple methods do not scale linearly.
3	Slow. Basic methods cannot go much beyond $N \approx 10^7$.

Curse of dimensionality

Algorithms involving rank-structured matrices scale *very* poorly with dimension.

For instance, for the classical Fast Multipole Method, key quantities scale as:

Dimension	Typical ranks	Number of "neighbors"	Length of "interaction list"
1	2	2	3
2	10–50	8	27
3	50–500	26	189
d	$(\log(1/\varepsilon))^{d-1}$	$3^d - 1$	$6^d - 3^d$

For fast direct solvers, the scaling with dimension is equally problematic:

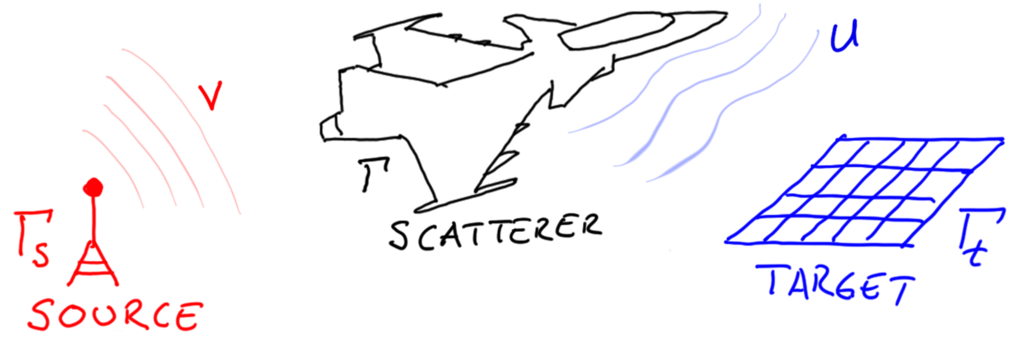
Effective Dimension	Current state of affairs
1	Extremely fast. Linear scaling is easy to attain.
2	Quite fast in practice, but simple methods do not scale linearly.
3	Slow. Basic methods cannot go much beyond $N \approx 10^7$.

Fortunately, we can often reduce the *effective* dimensionality.

For many 3D problems, the dense problems we need to invert "live" on 2D domains!

Curse of dimensionality: Dimension reduction via an integral equation

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:



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

The BVP (3) has an alternative mathematical formulation in the BIE

$$(4) \qquad -\pi i \sigma(\boldsymbol{x}) + \int_{\partial\Omega} \left(\left(\partial_{\boldsymbol{n}(\boldsymbol{y})} + i\kappa\right) \frac{e^{i\kappa|\boldsymbol{x}-\boldsymbol{y}|}}{|\boldsymbol{x}-\boldsymbol{y}|} \right) \sigma(\boldsymbol{y})\, dS(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega.$$

The integral equation (4) has several advantages over the PDE (3), including:

- The domain of computation $\partial\Omega$ is finite.
- The domain of computation $\partial\Omega$ is 2D, while $\mathbb{R}^3\backslash\overline{\Omega}$ is 3D.
- Equation (4) is inherently well-conditioned (as a "2^{nd} kind Fredholm equation").

A serious drawback of integral equations is that they lead to *dense coefficient matrices.* Since we are interested in constructing inverses anyway, this is unproblematic for us!

Curse of dimensionality: Dimension reduction via sparse direct solvers

Let us next consider what happens if we directly discretize the PDE (using, say, finite elements or finite differences) to obtain a linear system

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

involving a *sparse* coefficient matrix **A**.

Key idea: Do a sparse LU factorization based on a "nested dissection" ordering of the grid as an outer solver. Then use rank structured matrix algebra to deal with the dense matrices that arise.

Curse of dimensionality: Dimension reduction via sparse direct solvers

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

$$(5) \qquad \begin{cases} -\Delta u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

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

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

where $[\mathbf{Au}](k) = \frac{1}{h^2}\big(4\,\mathbf{u}(k) - \mathbf{u}(k_\mathrm{s}) - \mathbf{u}(k_\mathrm{e}) - \mathbf{u}(k_\mathrm{n}) - \mathbf{u}(k_\mathrm{w})\big)$, see Figure B.

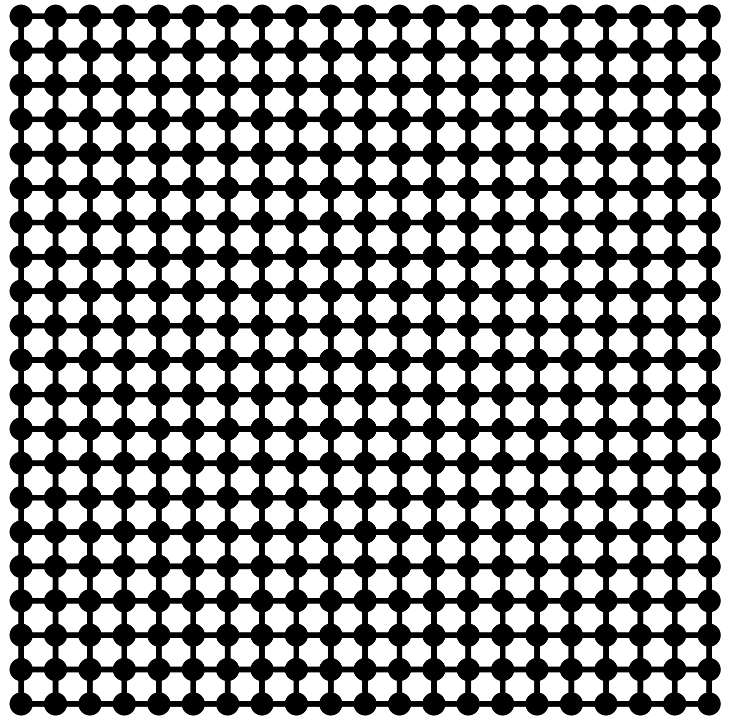


Figure A: The grid

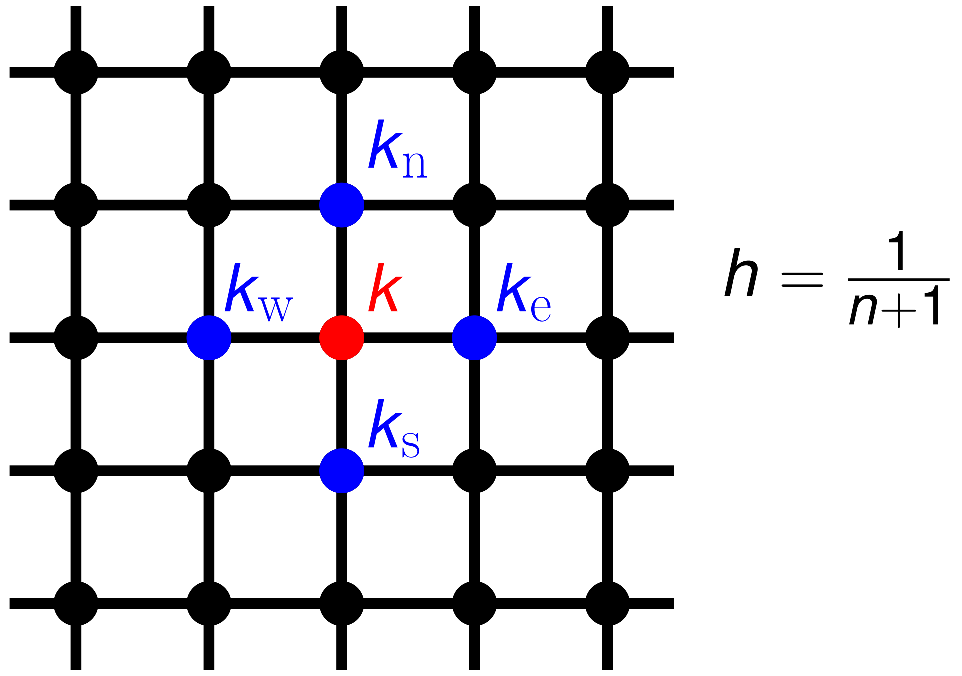
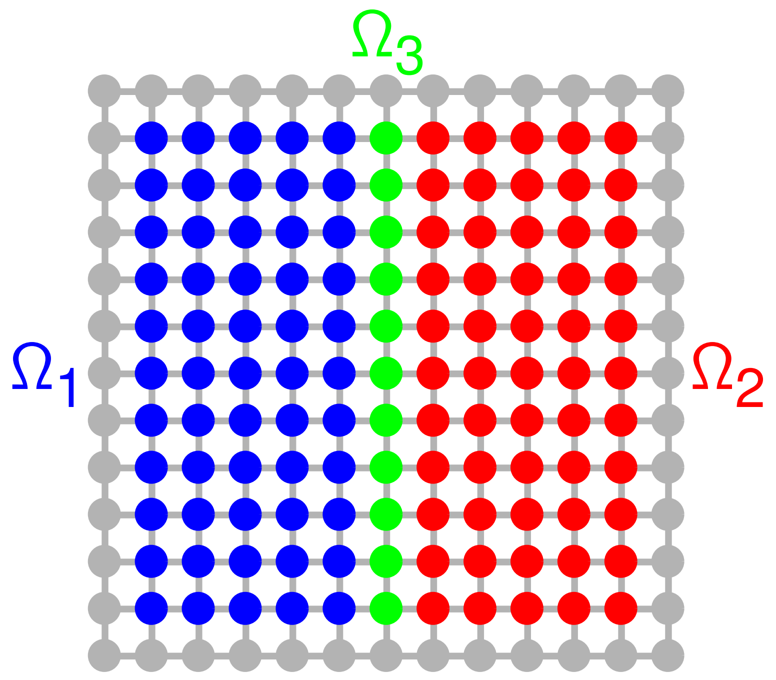


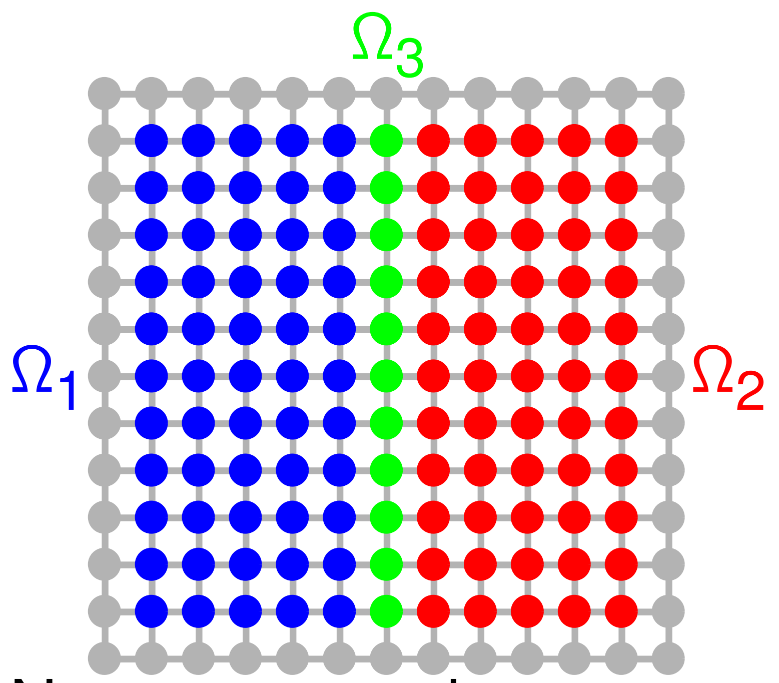
Figure B: The 5-point stencil

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:



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

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:



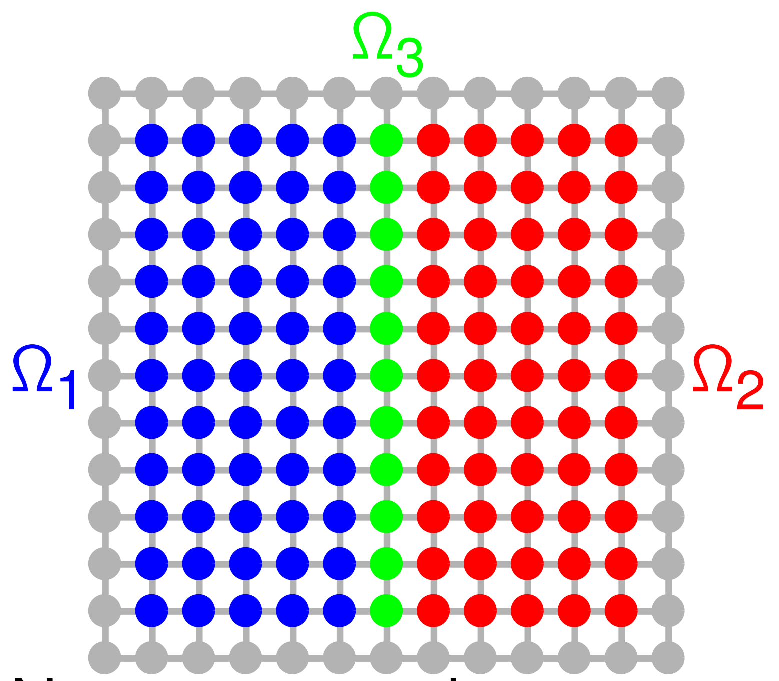
$$\mathbf{A} = \left[\begin{array}{c|c|c} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \hline \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \hline \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{array}\right]$$

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

$$\mathbf{A} = \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \hline \mathbf{A}_{31}\mathbf{A}_{11}^{-1} & \mathbf{A}_{32}\mathbf{A}_{22}^{-1} & \mathbf{I} \end{array}\right] \left[\begin{array}{c|c|c} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{array}\right] \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{A}_{13} \\ \hline \mathbf{0} & \mathbf{I} & \mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right]$$

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:



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

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

$$\mathbf{A} = \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \hline \mathbf{A}_{31}\mathbf{A}_{11}^{-1} & \mathbf{A}_{32}\mathbf{A}_{22}^{-1} & \mathbf{I} \end{array}\right] \left[\begin{array}{c|c|c} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{array}\right] \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{A}_{13} \\ \hline \mathbf{0} & \mathbf{I} & \mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right]$$

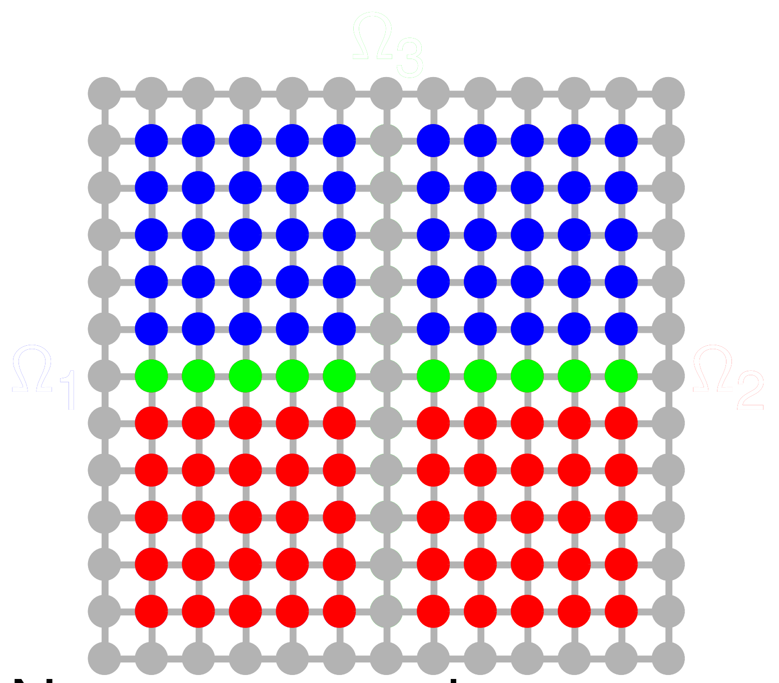
where $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$ is a *Schur complement.*

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:



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

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

$$\mathbf{A} = \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \hline \mathbf{A}_{31}\mathbf{A}_{11}^{-1} & \mathbf{A}_{32}\mathbf{A}_{22}^{-1} & \mathbf{I} \end{array}\right] \left[\begin{array}{c|c|c} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} \end{array}\right] \left[\begin{array}{c|c|c} \mathbf{I} & \mathbf{0} & \mathbf{A}_{11}^{-1}\mathbf{A}_{13} \\ \hline \mathbf{0} & \mathbf{I} & \mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{I} \end{array}\right]$$

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!

Sparse direct solvers with nested dissection ordering

Typically, nested dissection orderings are more complicated:

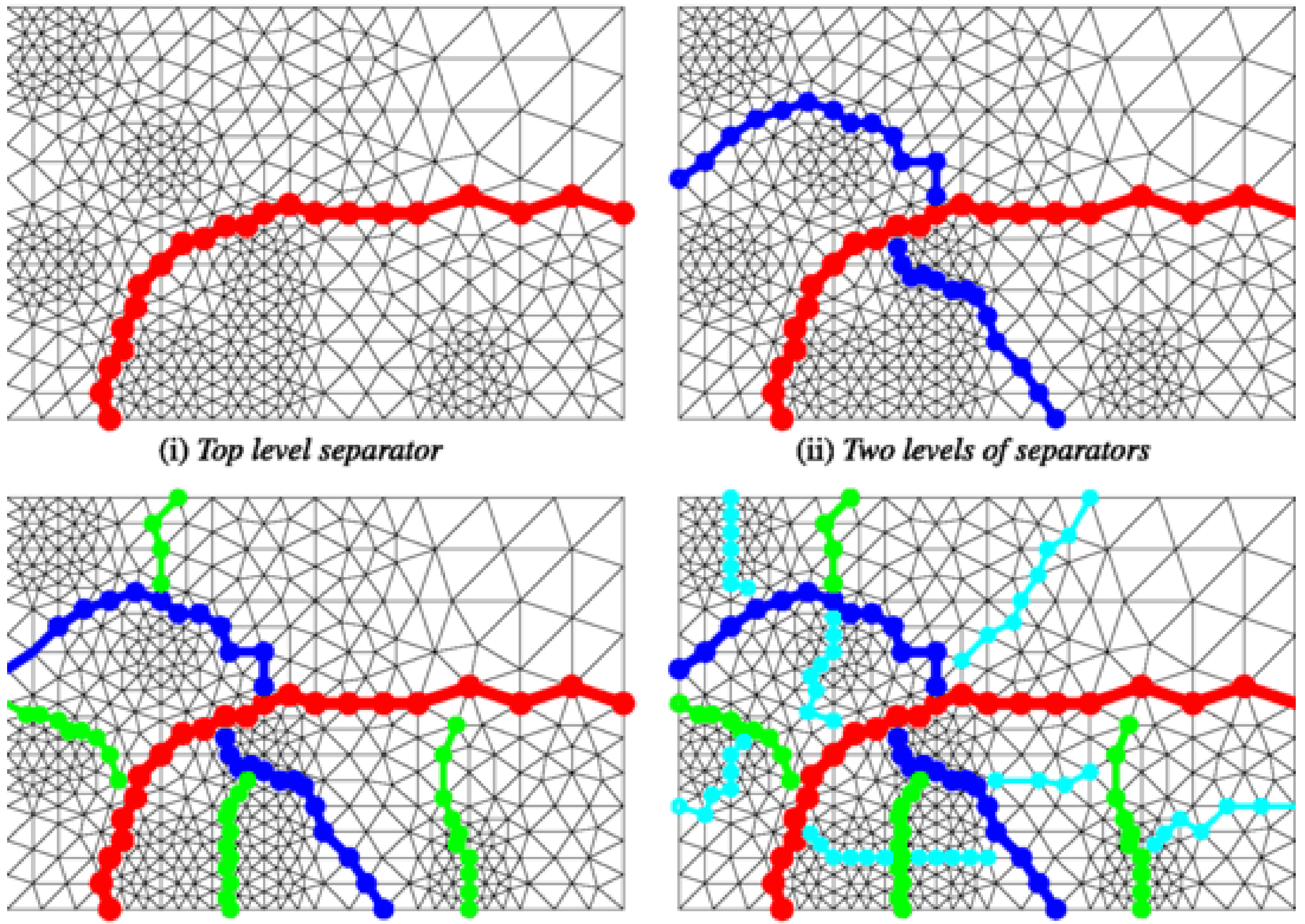

(i) *Top level separator*
(ii) *Two levels of separators*
(iii) *Three levels of separators*
(iv) *Four levels of separators*

Image credit: Jianlin Xia, "Robust and Efficient Multifrontal Solver for Large Discretized PDEs", 2012

Observe that while the computational domain is 2D in this example, the rank structured matrices all live on the colored 1D domains.

Curse of dimensionality: Dimension reduction

Key point: *When faced with a BVP in 3D, you can in most circumstances build direct solvers that rely only on dense operators associated with 2D domains.*

1. Constant coefficient problems: Reformulate as integral equation on boundary.
2. Variable coefficient problems: Use a sparse direct solver as an “outer” solver.

Outline of talk:

- Introduction: Problem formulation & solution operators. *[Done!]*
- Curse of dimensionality. *[Done!]*
- Interaction ranks — why are they small? How small are they?
- (Versions of fast direct solvers — “strong” versus “weak” etc.)
- High order discretizations and fast direct solvers.
- Randomized low rank approximation.
- Randomized compression of rank structured matrices.

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

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

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*.

Interaction ranks: Boundary integral equations

Let us consider two simple boundary integral equations on a boundary Γ:

The first is a reformulation of a Dirichlet problem involving the Laplace equation:

$$\alpha\sigma(\boldsymbol{x}) + \int_\Gamma \big(d(\boldsymbol{x},\boldsymbol{y}) + s(\boldsymbol{x},\boldsymbol{y})\big)\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

The second is a reformulation of a Dirichlet problem involving the Helmholtz equation:

$$\beta\sigma(\boldsymbol{x}) + \int_\Gamma \big(d_\kappa(\boldsymbol{x},\boldsymbol{y}) + i\kappa s_\kappa(\boldsymbol{x},\boldsymbol{y})\big)\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

The kernels are derived from the corresponding fundamental solutions:

$$\begin{aligned} s(\boldsymbol{x},\boldsymbol{y}) =& \phi(\boldsymbol{x}-\boldsymbol{y}),\\ d(\boldsymbol{x},\boldsymbol{y}) =& \partial_{\boldsymbol{n}(\boldsymbol{y})}\phi(\boldsymbol{x}-\boldsymbol{y}),\\ s_\kappa(\boldsymbol{x},\boldsymbol{y}) =& \phi_\kappa(\boldsymbol{x}-\boldsymbol{y}),\\ d_\kappa(\boldsymbol{x},\boldsymbol{y}) =& \partial_{\boldsymbol{n}(\boldsymbol{y})}\phi_\kappa(\boldsymbol{x}-\boldsymbol{y}), \end{aligned}$$

where, as before,

$$\begin{aligned} \phi(\boldsymbol{x}) =& -\frac{1}{2\pi}\log|\boldsymbol{x}|,\\ \phi_\kappa(\boldsymbol{x}) =& \frac{i}{4}H_0^{(1)}(\kappa|\boldsymbol{x}|). \end{aligned}$$

Interaction ranks: Boundary integral equations

Let us consider two simple boundary integral equations on a boundary Γ:

The first is a reformulation of a Dirichlet problem involving the Laplace equation:

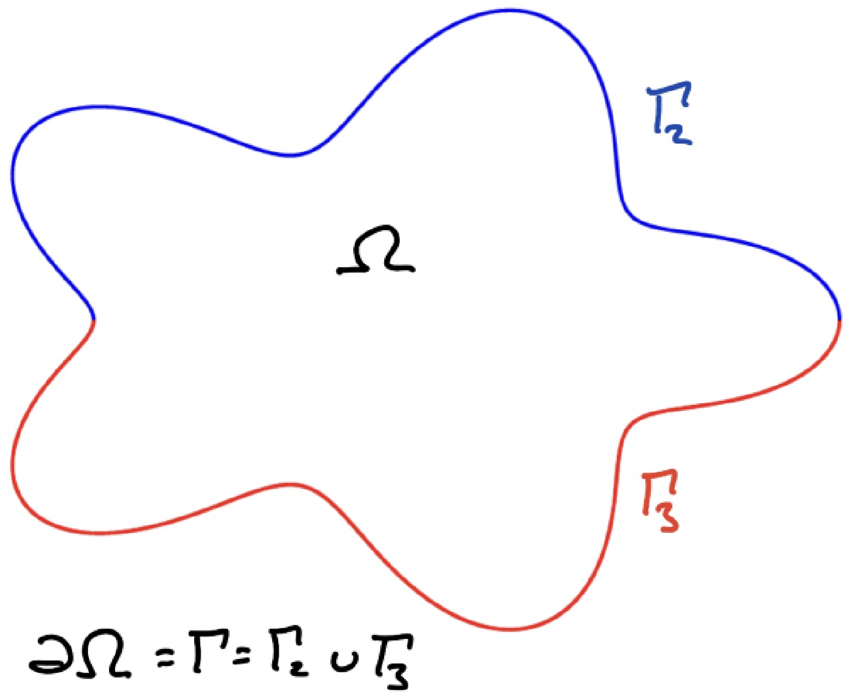
$$\alpha\sigma(\boldsymbol{x}) + \int_\Gamma \big(d(\boldsymbol{x},\boldsymbol{y}) + s(\boldsymbol{x},\boldsymbol{y})\big)\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

The second is a reformulation of a Dirichlet problem involving the Helmholtz equation:

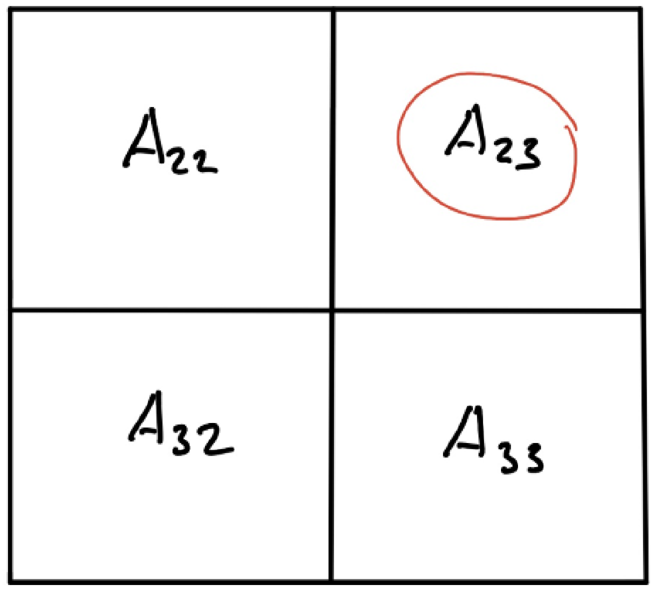
$$\beta\sigma(\boldsymbol{x}) + \int_\Gamma \big(d_\kappa(\boldsymbol{x},\boldsymbol{y}) + i\kappa s_\kappa(\boldsymbol{x},\boldsymbol{y})\big)\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

Let **A** denote the matrix resulting from discretization of either BIE.

The geometry



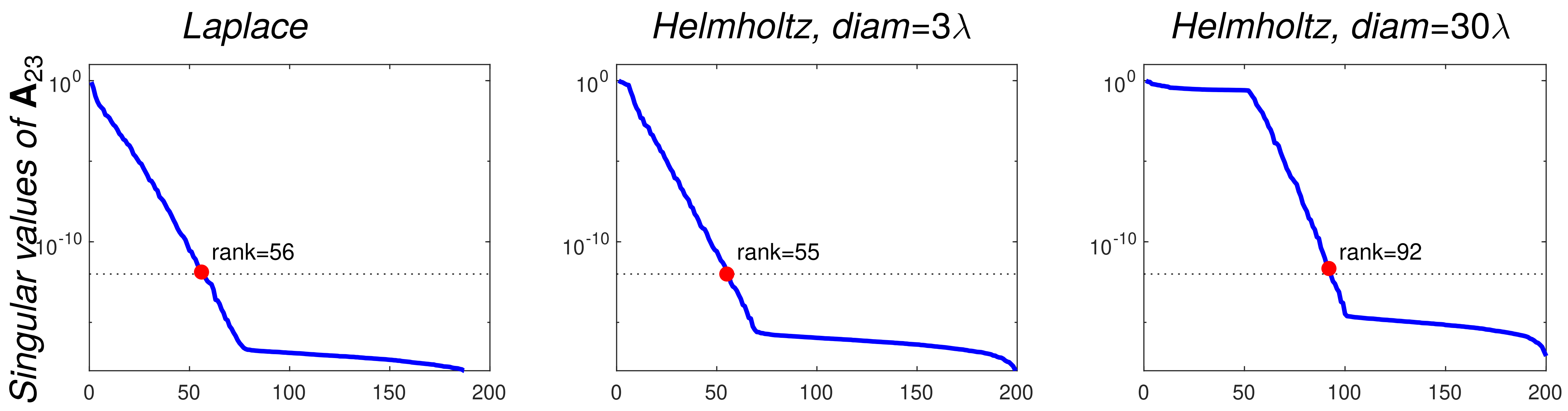
The matrix **A**



On the next slide, we show the singular values of the off-diagonal block $\mathbf{A}_{23}$.

Interaction ranks: Boundary integral equations

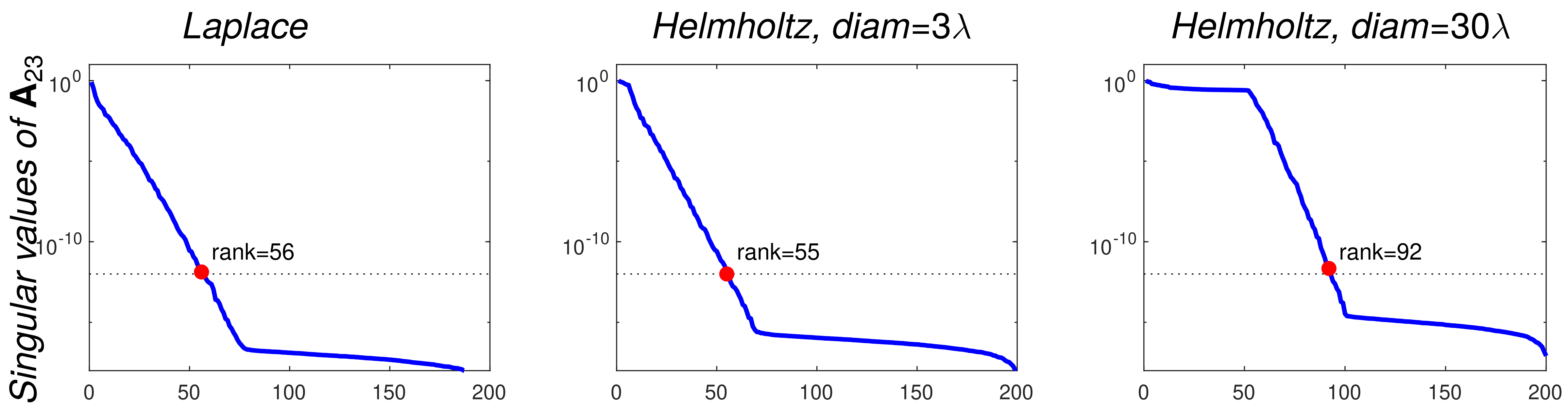
The ranks of an off-diagonal block of **A**:



This is all as expected. Somewhat accessible by analysis.

Interaction ranks: Boundary integral equations

The ranks of an off-diagonal block of **A**:

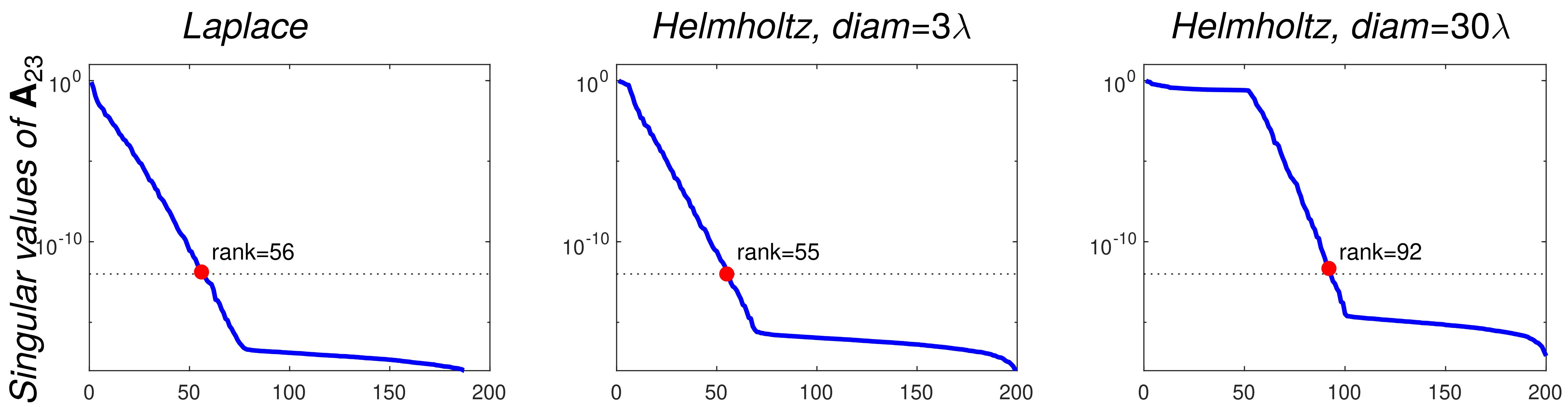


This is all as expected. Somewhat accessible by analysis.

Now the fun part! We set $\mathbf{B} = \mathbf{A}^{-1}$, and plot the svds of the off-diagonal block $\mathbf{B}_{23}$.

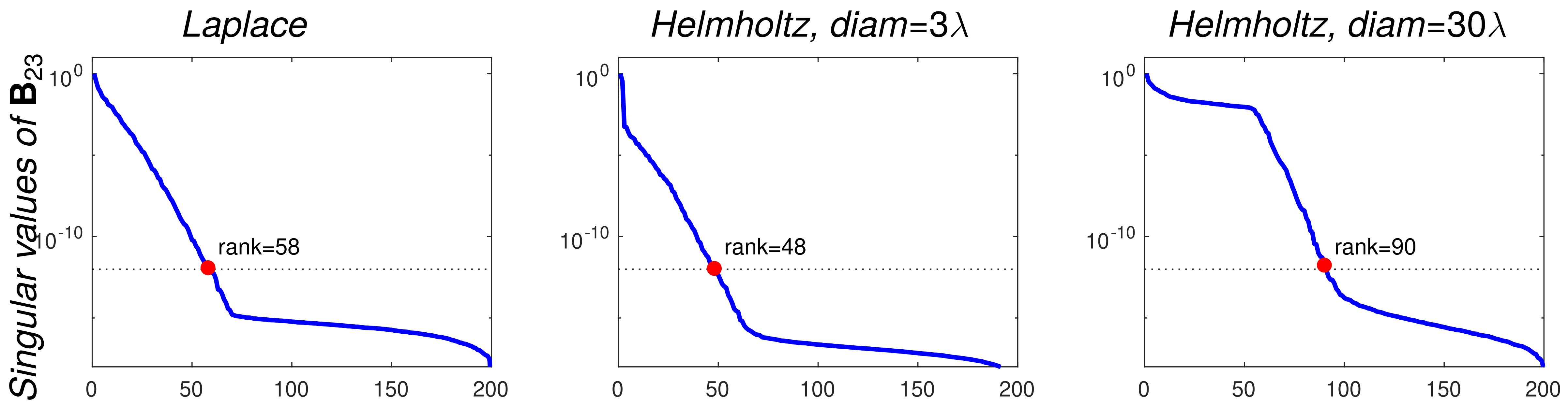
Interaction ranks: Boundary integral equations

The ranks of an off-diagonal block of **A**:



This is all as expected. Somewhat accessible by analysis.

Now the fun part! We set $\mathbf{B} = \mathbf{A}^{-1}$, and plot the svds of the off-diagonal block $\mathbf{B}_{23}$.

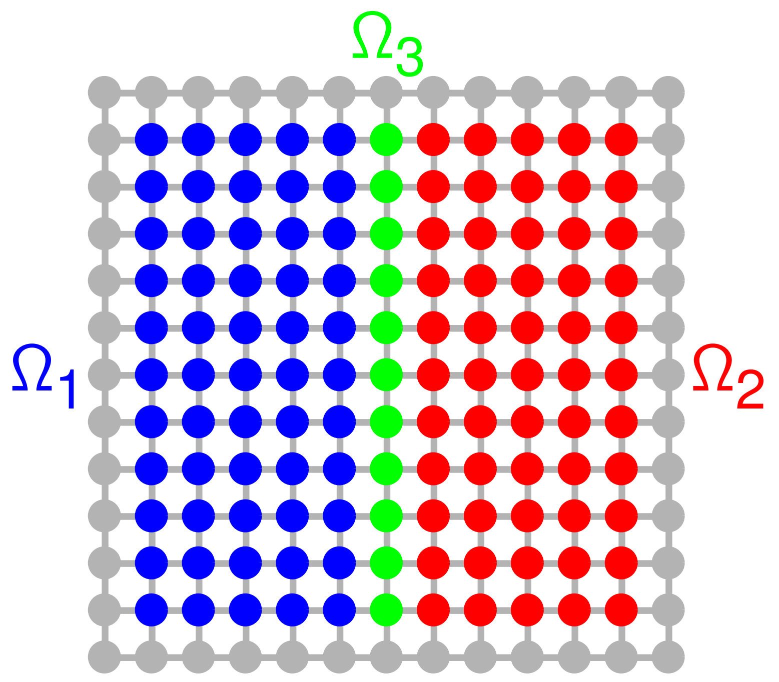


Remarkable similarity!

(Observe ill-conditioning due to close resonances for the Helmholtz BIE.)

Interaction ranks: Stiffness matrix from finite difference discretization

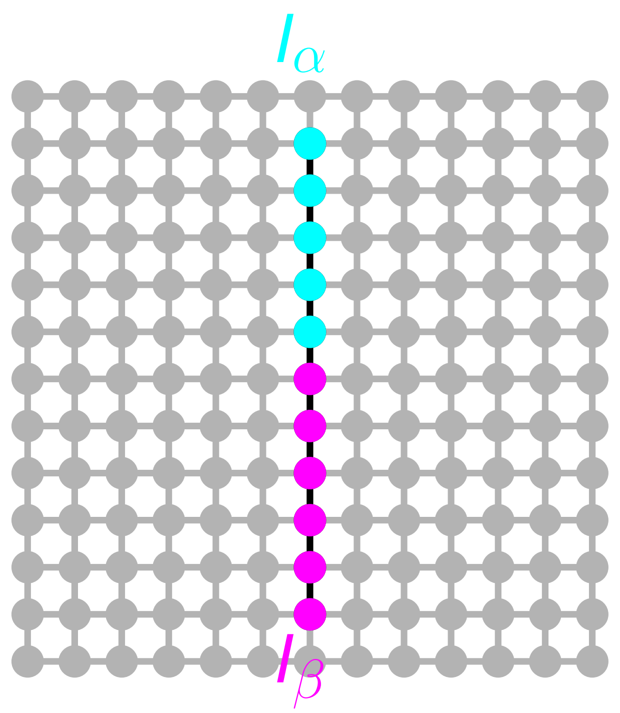
Recall our example of Laplace's equation discretized using the 5-point stencil.



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

We build the Schur complement $\mathbf{S} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$.

Then split the Schur complement into four parts:

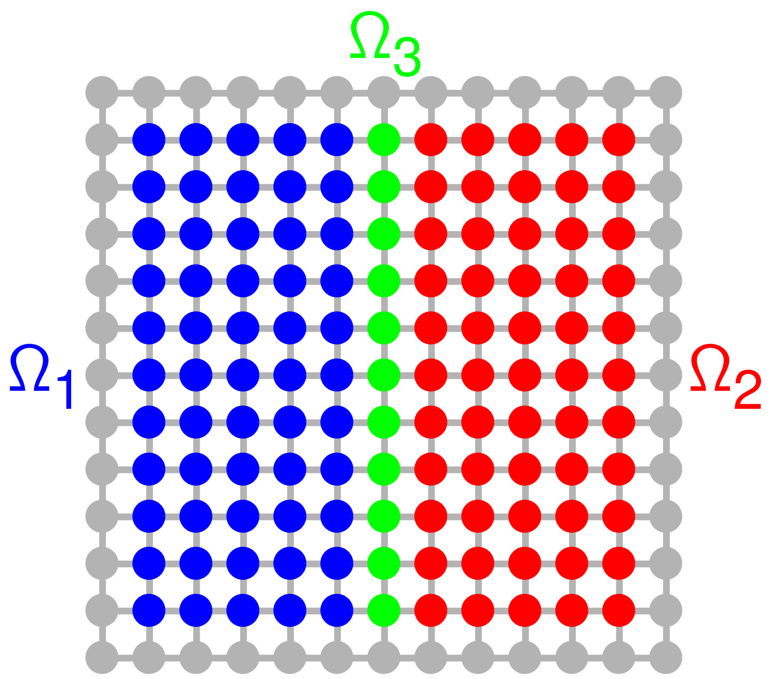


$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{\alpha\alpha} & \mathbf{S}_{\alpha\beta} \\ \mathbf{S}_{\beta\alpha} & \mathbf{S}_{\beta\beta} \end{bmatrix}$$

We explore the svds of $\mathbf{S}_{\alpha\beta}$ — encoding interactions between I_α and I_β.

Interaction ranks: Stiffness matrix from finite difference discretization

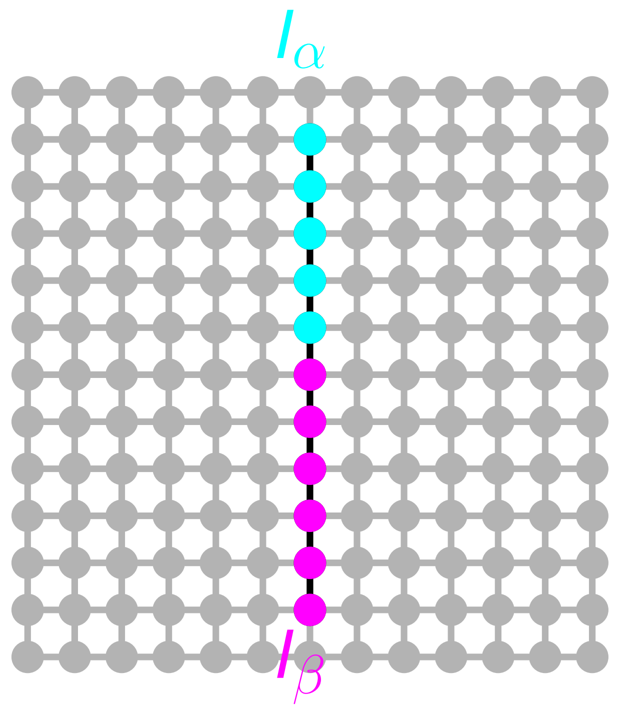
Recall our example of Laplace's equation discretized using the 5-point stencil.



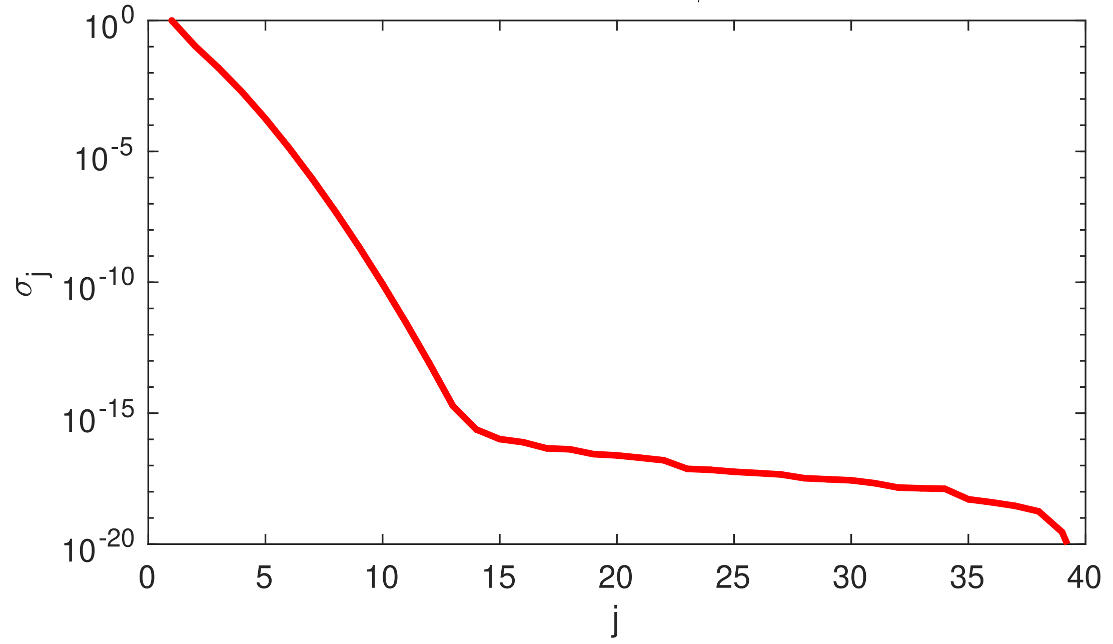
$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}$$

We build the Schur complement $\mathbf{S} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$.

Then split the Schur complement into four parts:



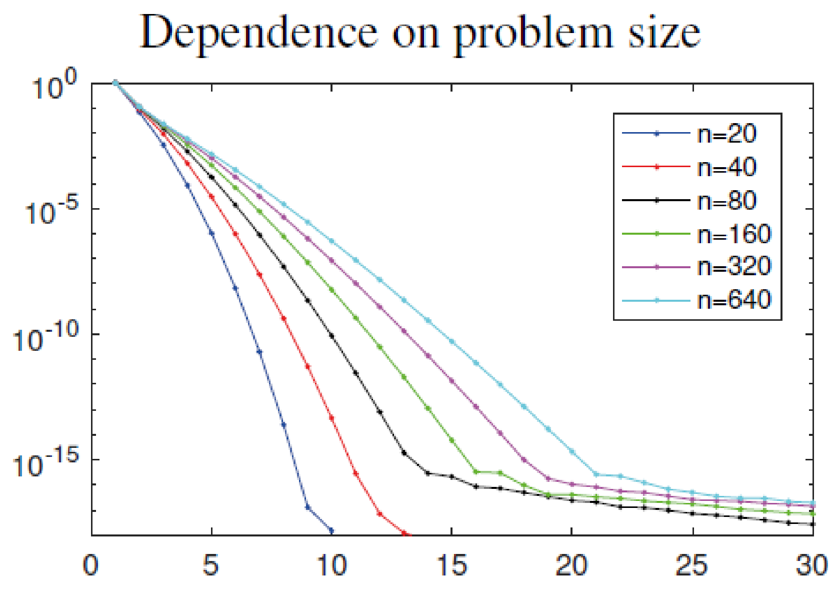
Singular values of $S_{\alpha\beta}$ for an 80×80 *grid.*



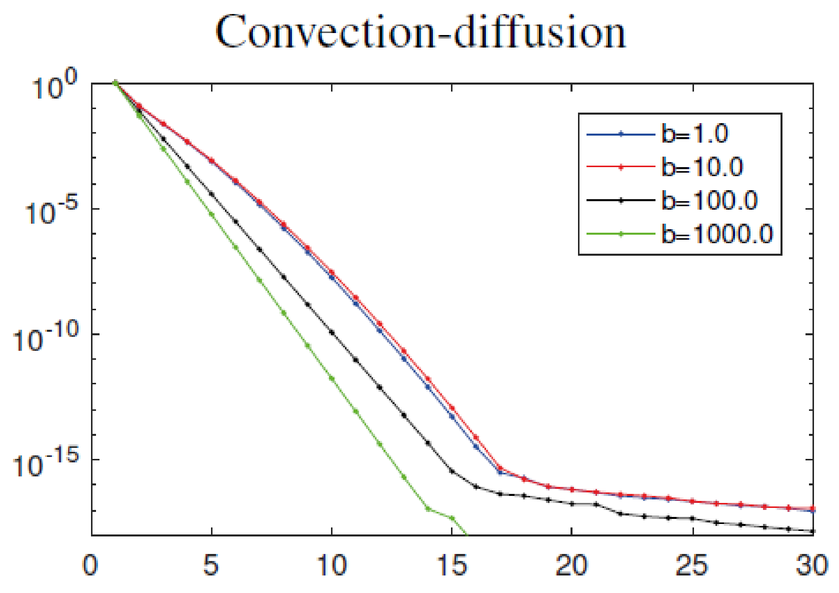
We explore the svds of $\mathbf{S}_{\alpha\beta}$ — encoding interactions between I_α and I_β.

Interaction ranks: Stiffness matrix from finite difference discretization

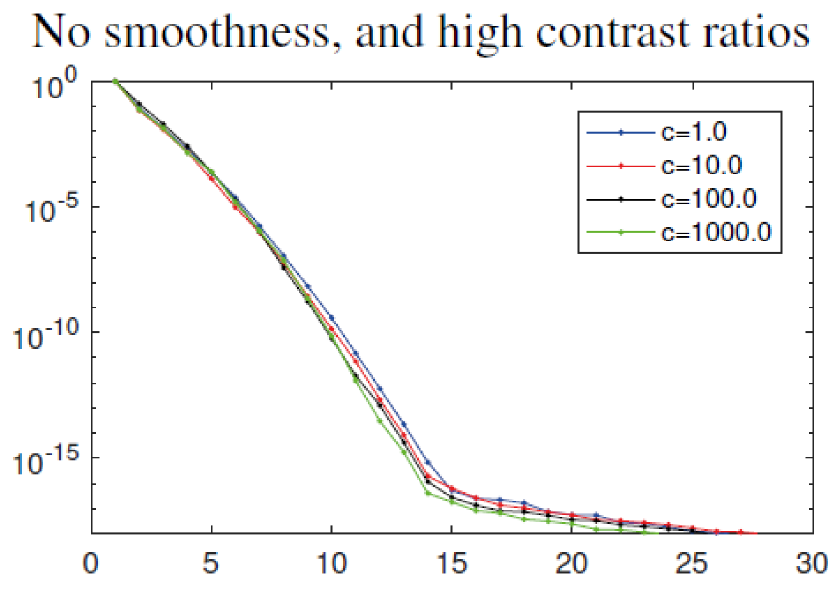
Let us try a few different PDEs, and different problem sizes:



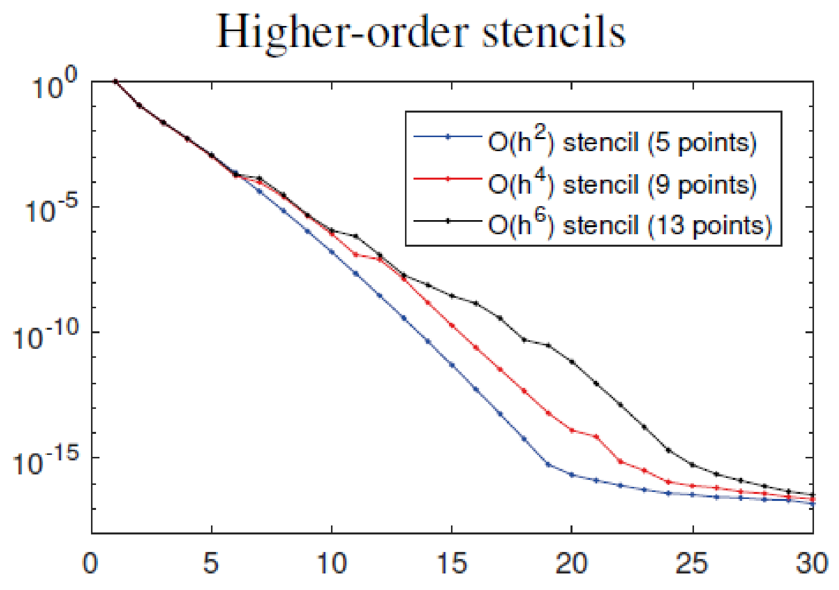
(a)



(b)



(c)



(d)

Note: The rank decay property is remarkably stable!

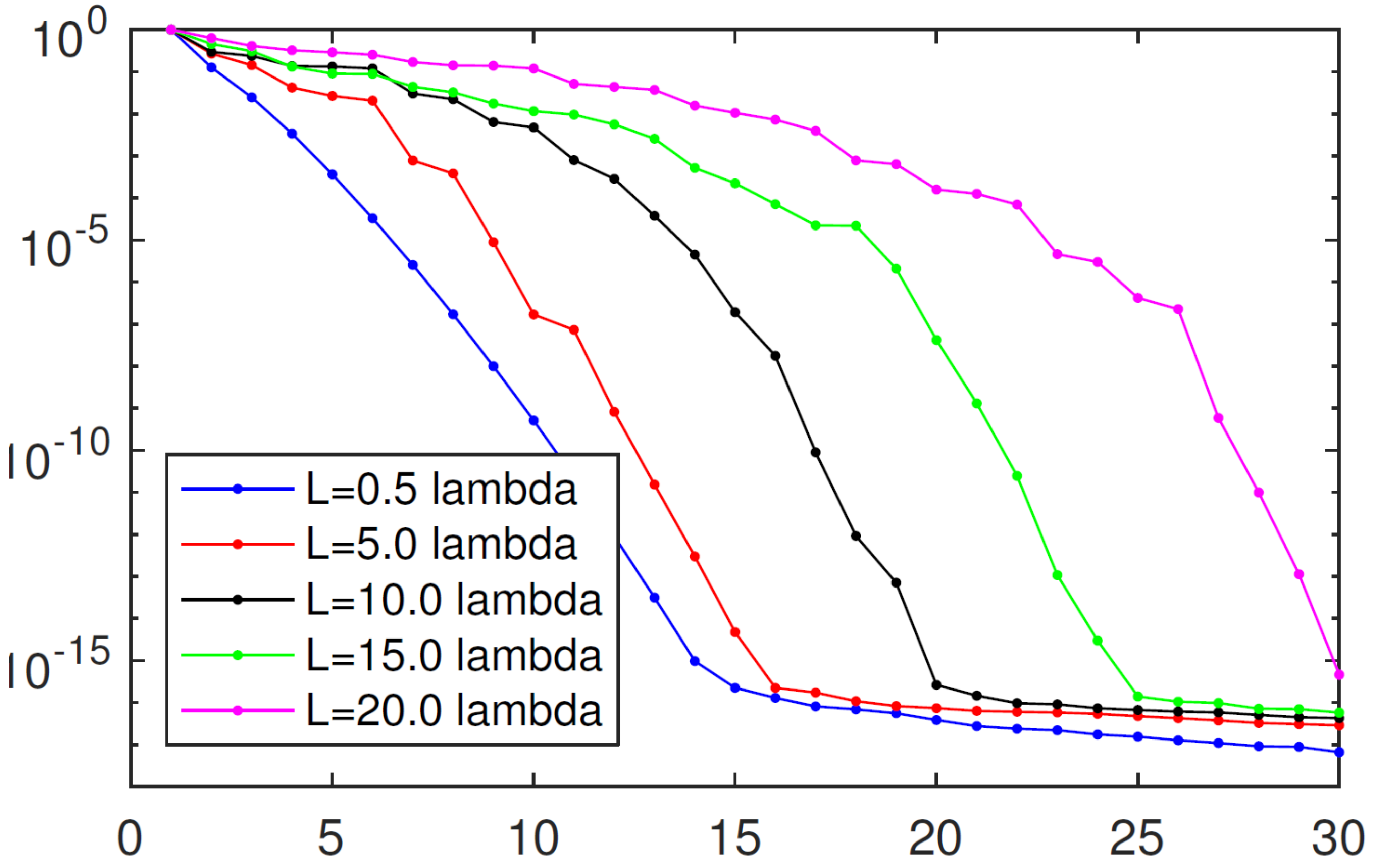
Note: The decay continues to ϵ_{mach} — regardless of the discretization errors!

Interaction ranks: Stiffness matrix from finite difference discretization

Next, let us consider Helmholtz problems with increasing wave numbers.

Interaction ranks: Stiffness matrix from finite difference discretization

Next, let us consider Helmholtz problems with increasing wave numbers.

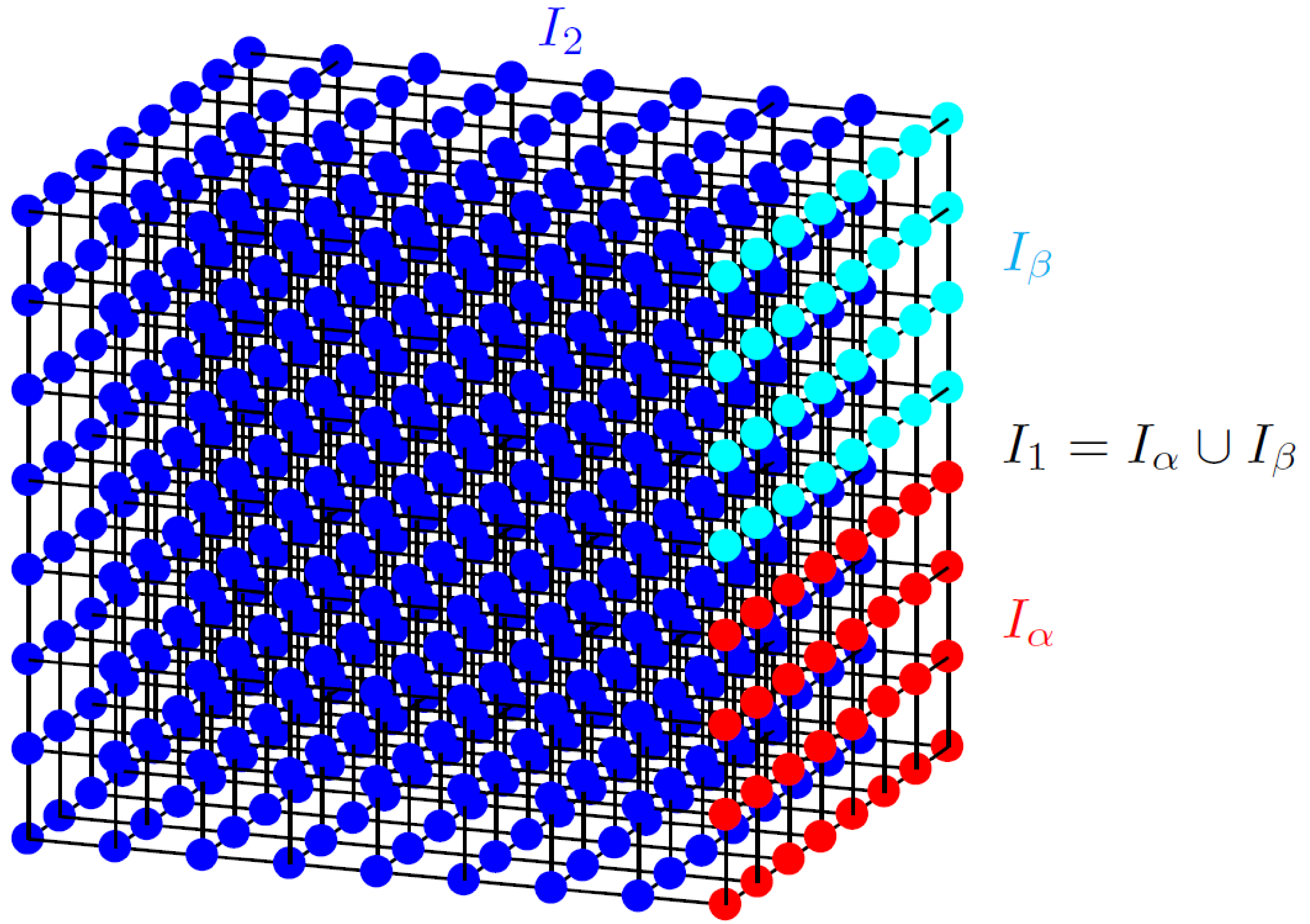


We recognize this pattern from the potential evaluation operator:

Fast decay *once oscillations are resolved.*

Interaction ranks: Stiffness matrix from finite difference discretization

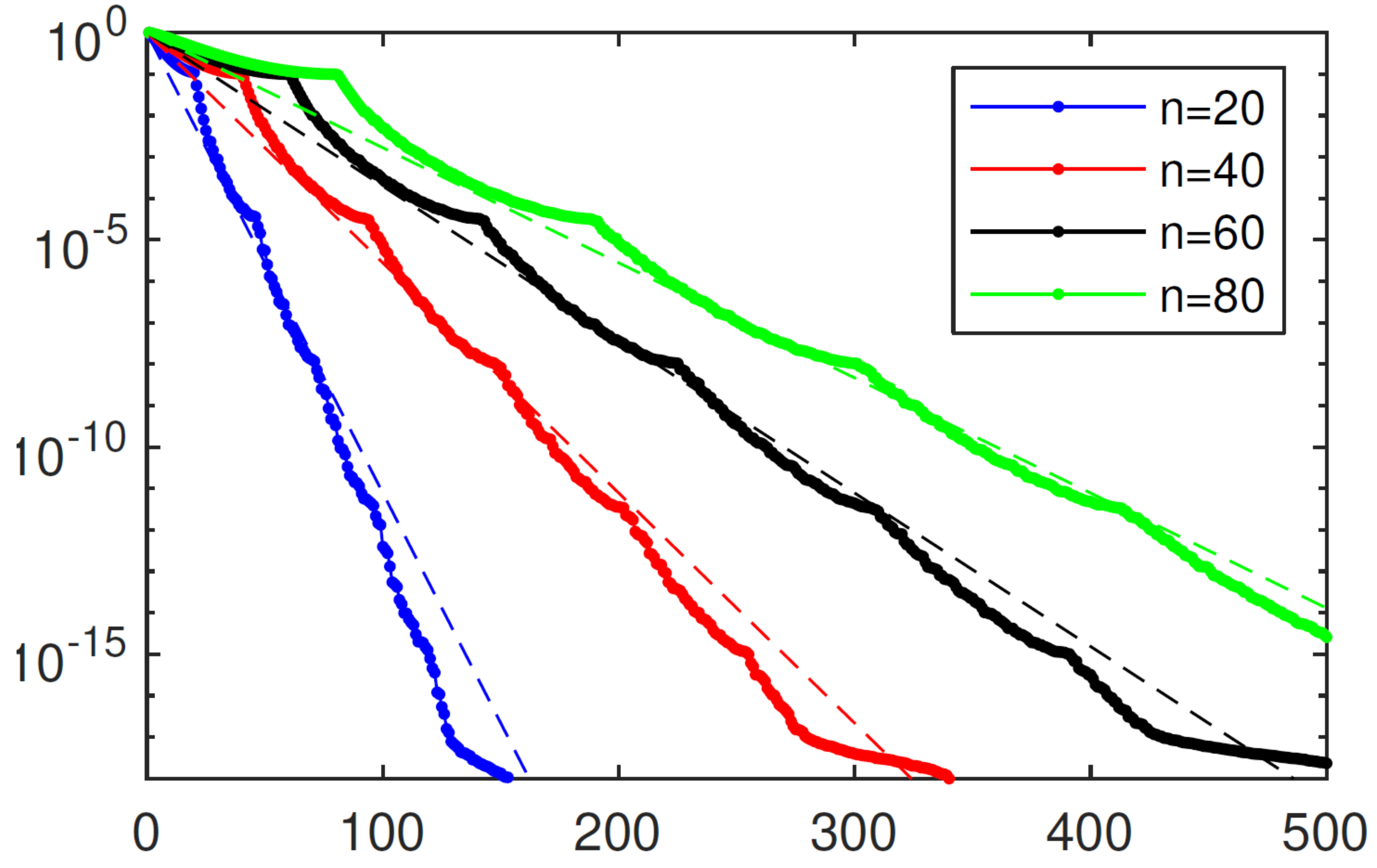
Finally, let us consider the analogous 3D problem.



The geometry.

Interaction ranks: Stiffness matrix from finite difference discretization

Finally, let us consider the analogous 3D problem.



The singular values.

Outline of talk:

- Introduction: Problem formulation & solution operators. *[Done!]*
- Curse of dimensionality. *[Done!]*
- Interaction ranks — why are they small? How small are they? *[Done!]*
- (Versions of fast direct solvers — “strong” versus “weak” etc.)
- High order discretizations and fast direct solvers.
- Randomized low rank approximation.
- Randomized compression of rank structured matrices.

Outline of talk:

- Introduction: Problem formulation & solution operators. *[Done!]*
- Curse of dimensionality. *[Done!]*
- Interaction ranks — why are they small? How small are they? *[Done!]*
- (Versions of fast direct solvers — “strong” versus “weak” etc.) *[Skipped.]*
- High order discretizations and fast direct solvers.
- Randomized low rank approximation.
- Randomized compression of rank structured matrices.

Fast direct solvers and high order methods

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

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

  $\rightarrow$ *You want to maximize the oomph per DOF.*
- Direct solvers are particularly well suited for medium frequency wave problems.

  $\rightarrow$ *Need high accuracy due to ill-conditioned physics.*
- High order methods sometimes lead to more ill-conditioned systems.

  $\rightarrow$ *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.

Fast direct solvers and high order methods

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

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

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

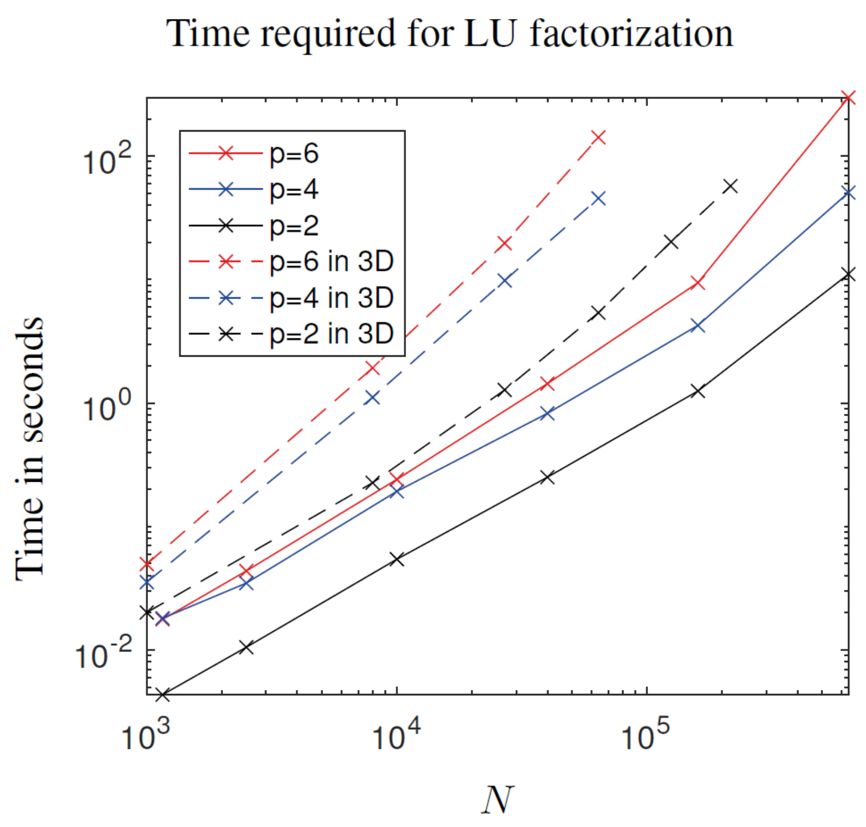
- Direct solvers are particularly well suited for medium frequency wave problems.

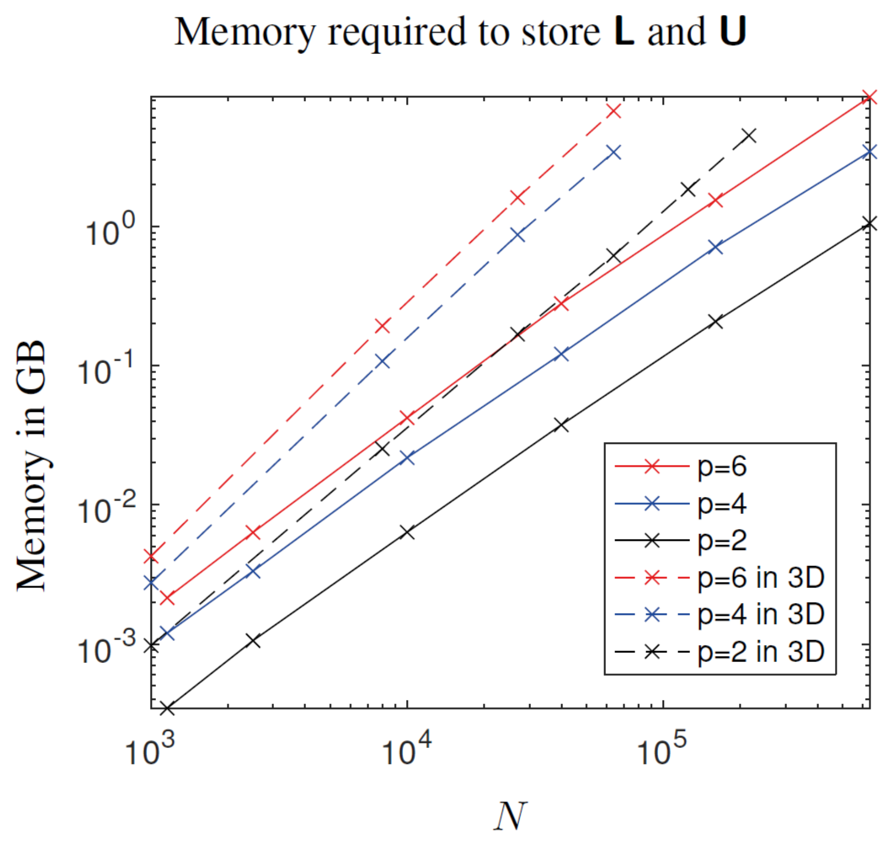
  $\rightarrow$ *Need high accuracy due to ill-conditioned physics.*

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

  $\rightarrow$ *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.





Fast direct solvers and high order methods

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

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

  $\rightarrow$ *You want to maximize the oomph per DOF.*
- Direct solvers are particularly well suited for medium frequency wave problems.

  $\rightarrow$ *Need high accuracy due to ill-conditioned physics.*
- High order methods sometimes lead to more ill-conditioned systems.

  $\rightarrow$ *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: Pick your discretization scheme carefully!

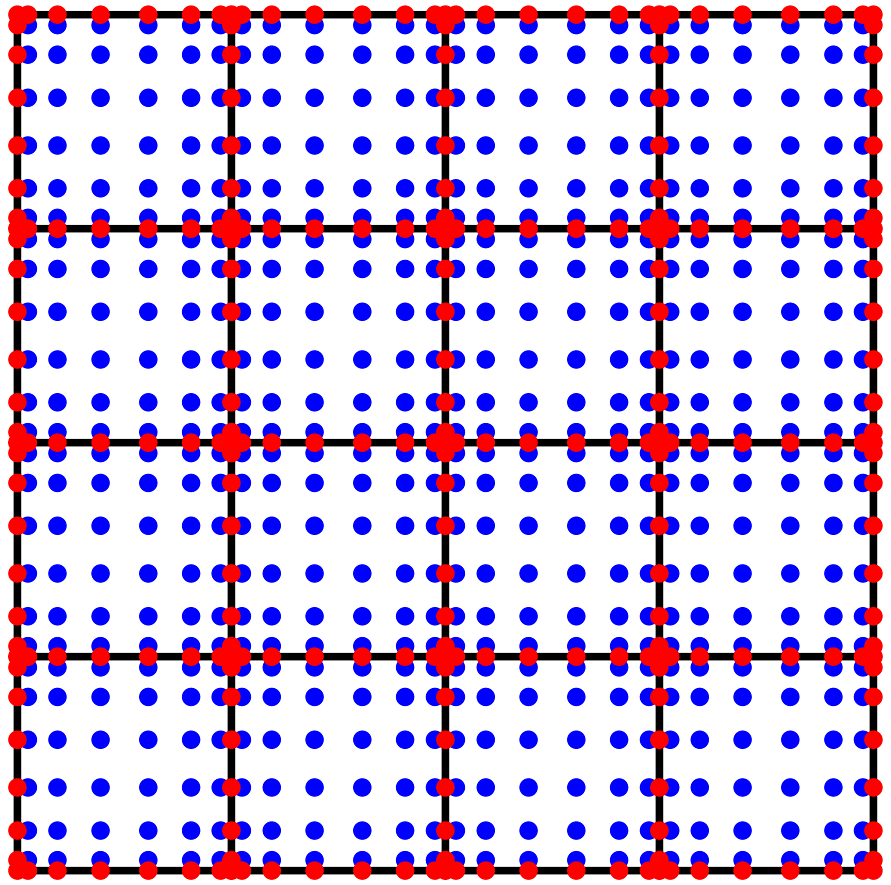
- When discretizing the PDE, use methods that play well with "static condensation". You want a clean separation between "interior" and "edge" degrees of freedom.
  - Multidomain spectral collocation methods ("HPS", "ultraSEM", etc.).
  - Discontinuous Galerkin. (Or so I speculate, at any rate.)
- When integral equation formulations are used, pick quadratures that have as localized "corrections" as possible. (Very technical point here!)

Fast direct solvers and high order methods: Multidomain spectral collocation

As a numerical illustration, let us consider the "Hierarchical Poincaré-Steklov (HPS)" method. We set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$, and consider the problem

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

We discretize using spectral collocation on a composite grid on Ω (Chebyshev nodes):



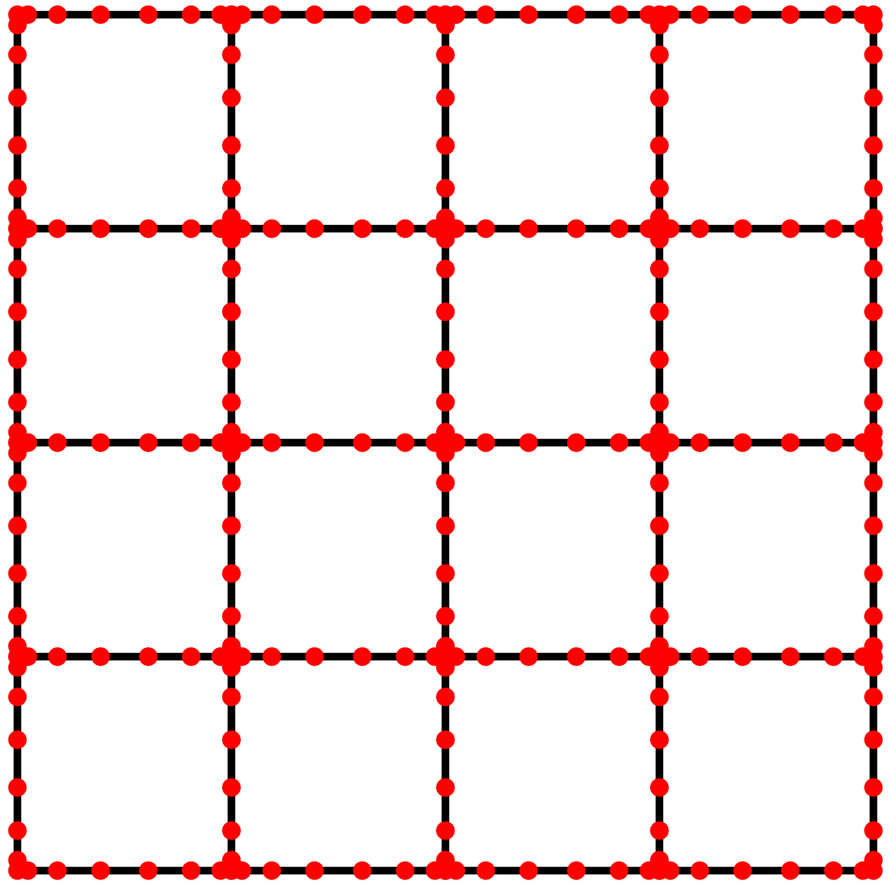
On patch boundaries, we enforce continuity of the potential and the normal derivative.

Fast direct solvers and high order methods: Multidomain spectral collocation

As a numerical illustration, let us consider the "Hierarchical Poincaré-Steklov (HPS)" method. We set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$, and consider the problem

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

We discretize using spectral collocation on a composite grid on Ω (Chebyshev nodes):



On patch boundaries, we enforce continuity of the potential and the normal derivative.

Fast direct solvers and high order methods: Multidomain spectral collocation

As a numerical illustration, let us consider the “Hierarchical Poincaré-Steklov (HPS)” method. We set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$, and consider the problem

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

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

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

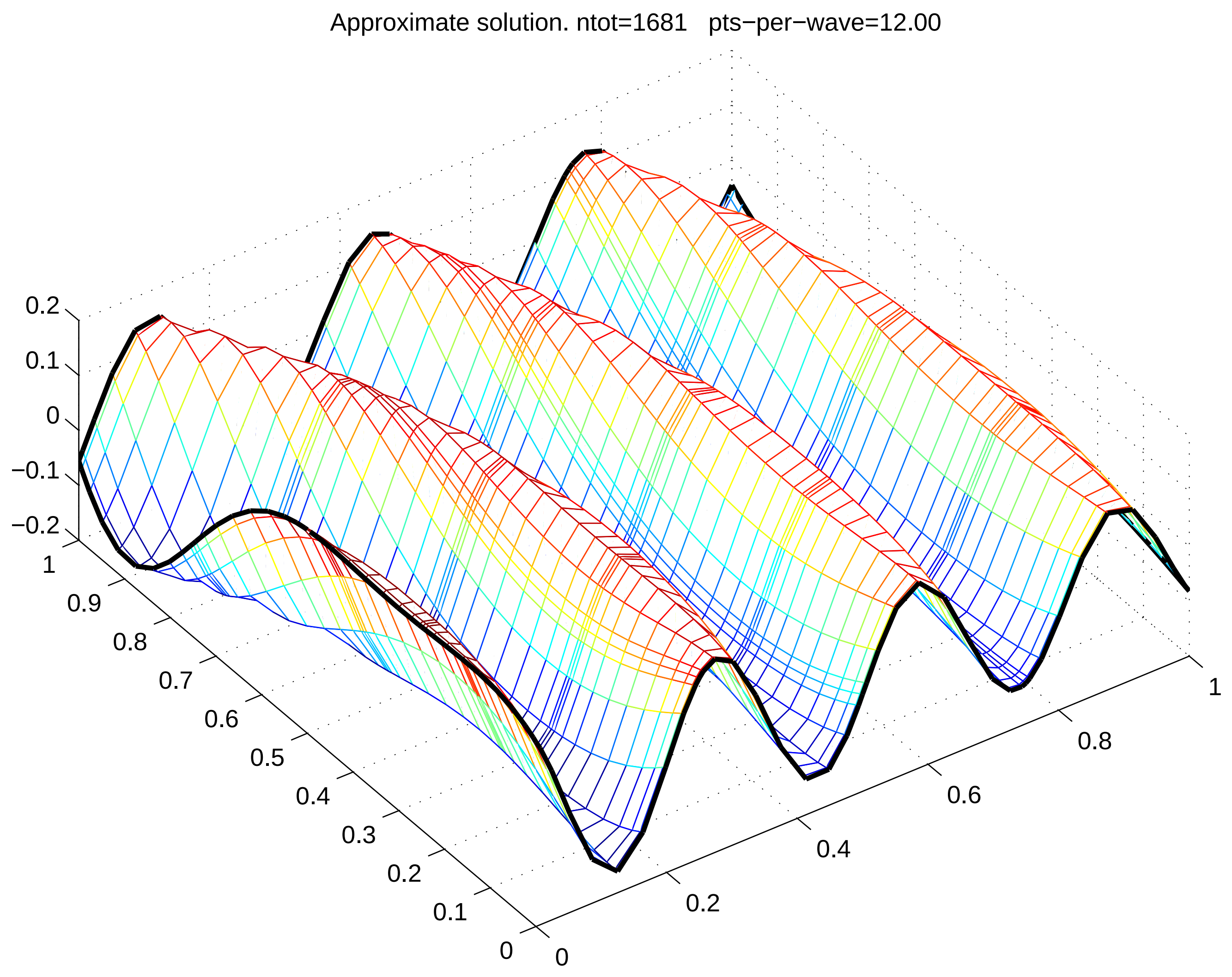
Fast direct solvers and high order methods: Multidomain spectral collocation

As a numerical illustration, let us consider the "Hierarchical Poincaré-Steklov (HPS)" method. We set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$, and consider the problem

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

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

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



Fast direct solvers and high order methods: Multidomain spectral collocation

As a numerical illustration, let us consider the “Hierarchical Poincaré-Steklov (HPS)” method. We set $\Omega = [0,1]^2$ and $\Gamma = \partial\Omega$, and consider the problem

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

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

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

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

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

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

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

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

Note: Fixed number of points per wave-length. Almost no “pollution error”!

Fast direct solvers and high order methods: Multidomain spectral collocation

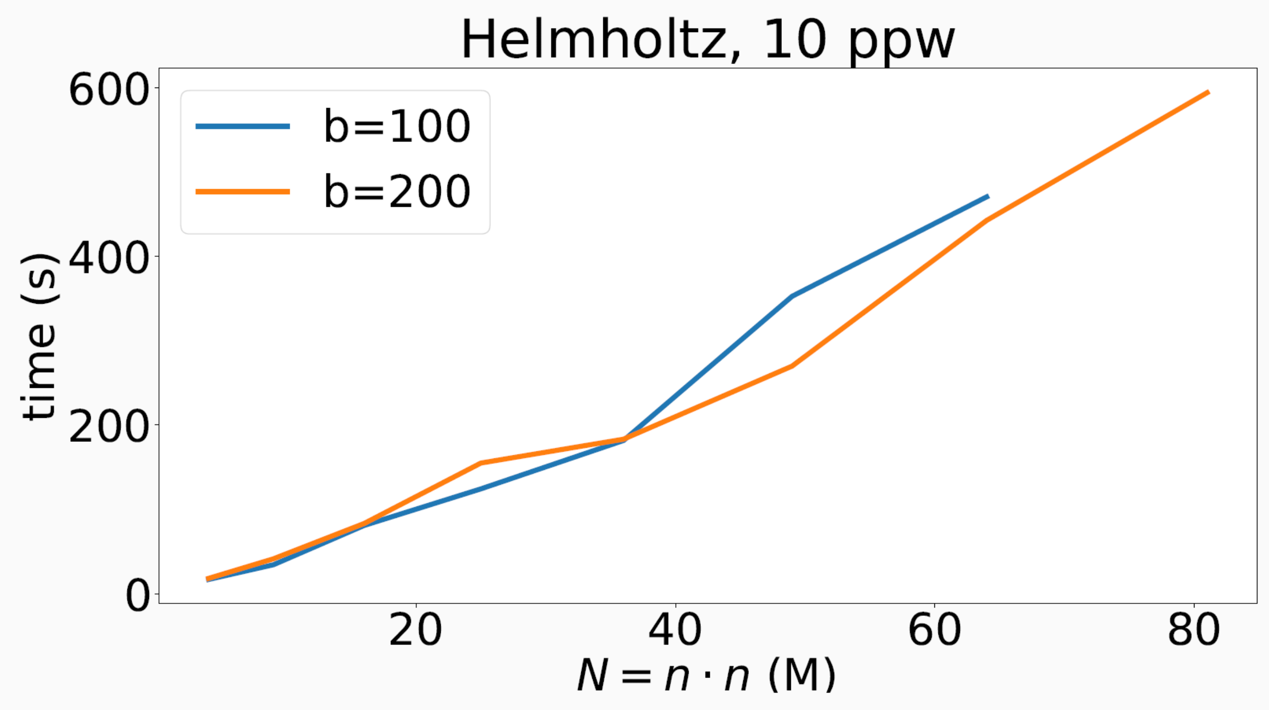
By incorporating rank-structured matrix algebra for the Schur complements, we can access larger problem sizes, and get linear scaling in important cases.

Problem	N	$T_{\rm build}$	$T_{\rm solve}$	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.)

Fast direct solvers and high order methods: Multidomain spectral collocation

Another set of numerical results. More recent. (With CSEM graduate Anna Yesypenko.)



b=100

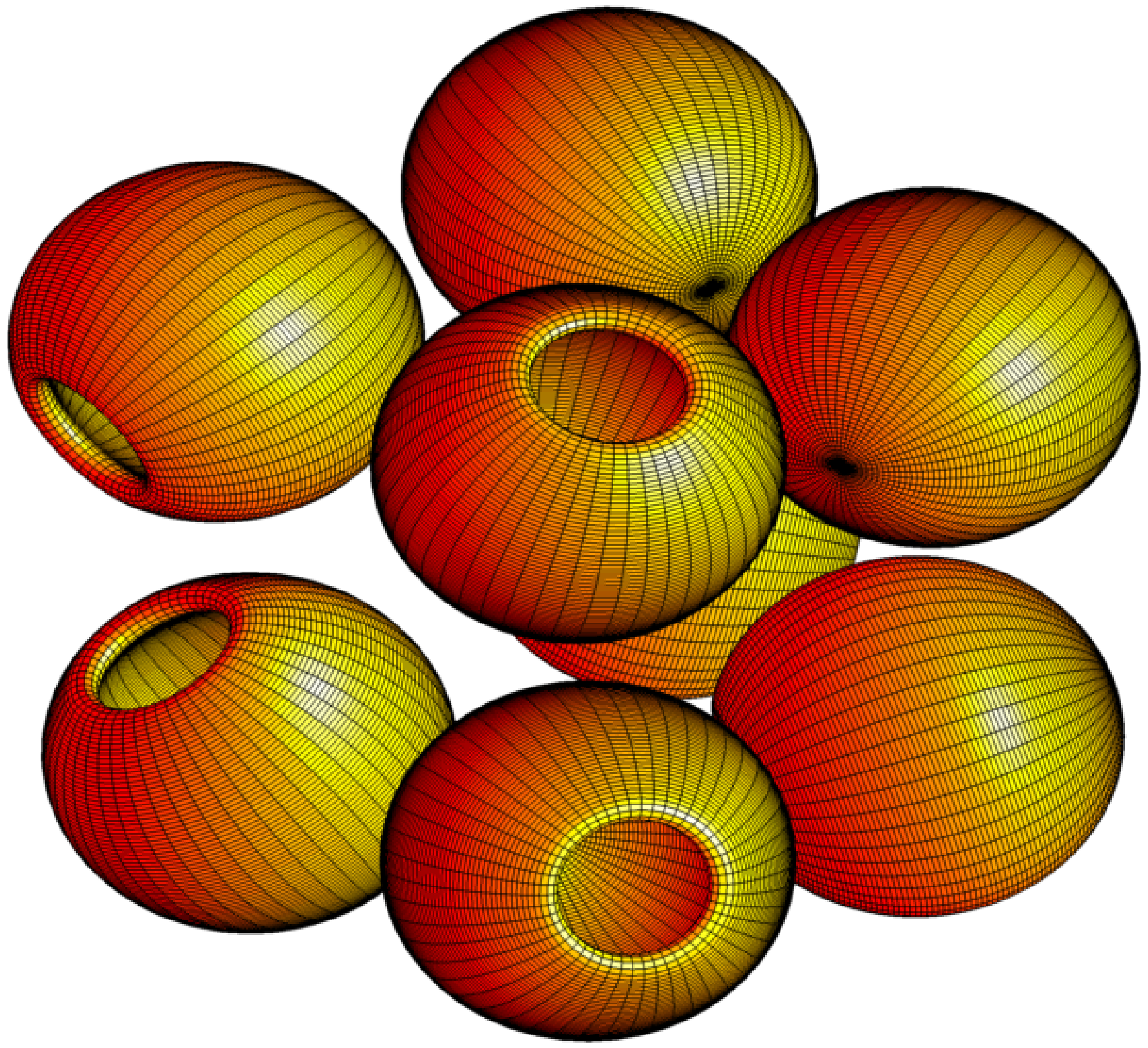
N	T build (s)	mem (GB)	T solve (s)	err
4.0 M	16.93	3.35	1.60	1.77e-08
9.0 M	34.36	8.09	3.39	5.15e-07
16.0 M	80.97	16.37	6.12	4.40e-06
25.0 M	124.32	27.35	9.45	1.63e-03
36.0 M	181.69	42.16	13.57	3.41e-06
49.0 M	352.66	64.15	19.09	1.04e-06
64.0 M	470.53	88.63	16.98	7.29e-07

b=200

N	T build (s)	mem (GB)	T solve (s)	err
4.0 M	18.21	4.05	1.19	4.65e-09
9.0 M	41.41	9.52	2.65	4.30e-06
16.0 M	83.35	17.83	4.67	4.67e-08
25.0 M	154.95	29.97	10.97	5.36e-07
36.0 M	183.25	42.32	16.11	1.06e-05
49.0 M	269.89	56.81	22.47	2.33e-03
64.0 M	442.62	83.21	30.79	3.10e-04
81.0 M	594.03	110.51	39.62	6.42e-04

Solving the Helmholtz equation using the HPS discretization scheme (10 points per wavelength, $p = 20$). The "medium complexity" option of low rank in the off-diagonal blocks of **T** *is deployed. Notice the high accuracy for a domain of size $900\lambda \times 900\lambda$.*

Let us consider a multibody scattering problem involving multiple cavities:

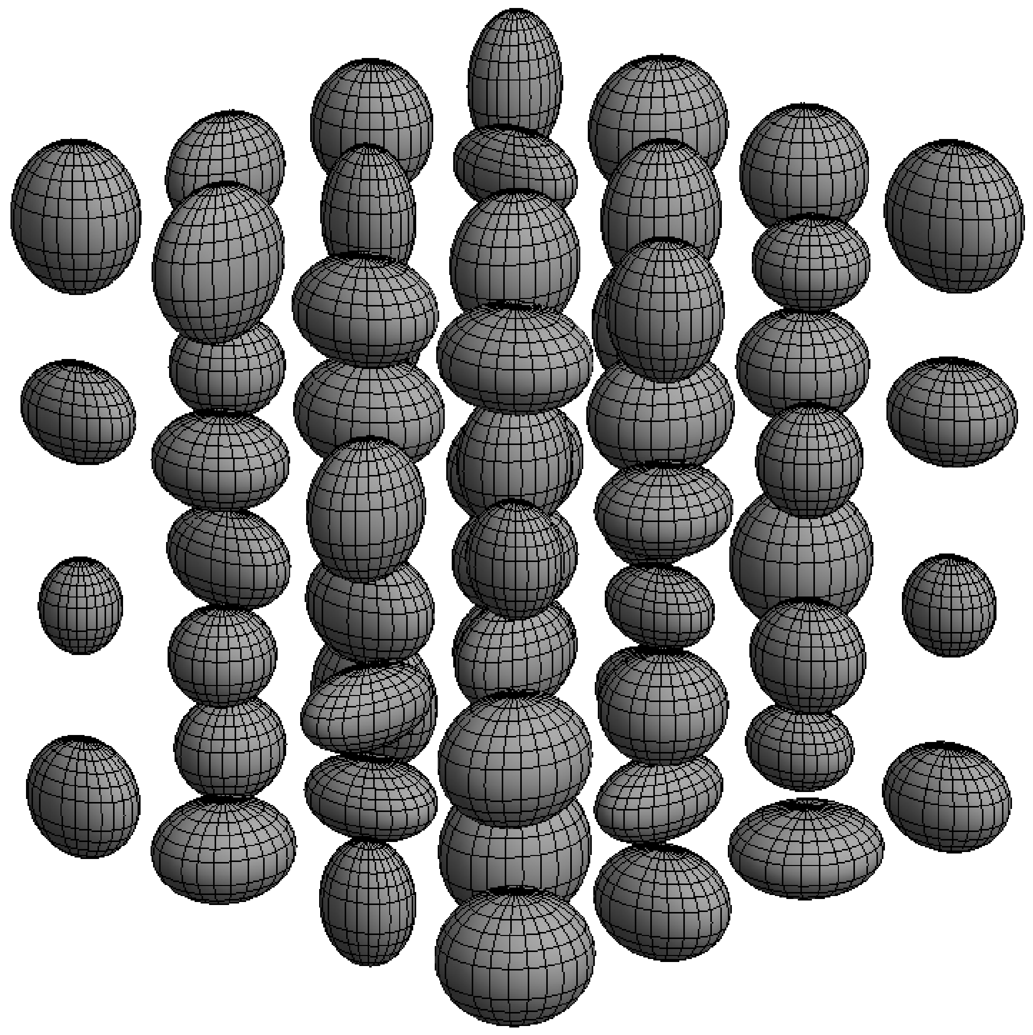
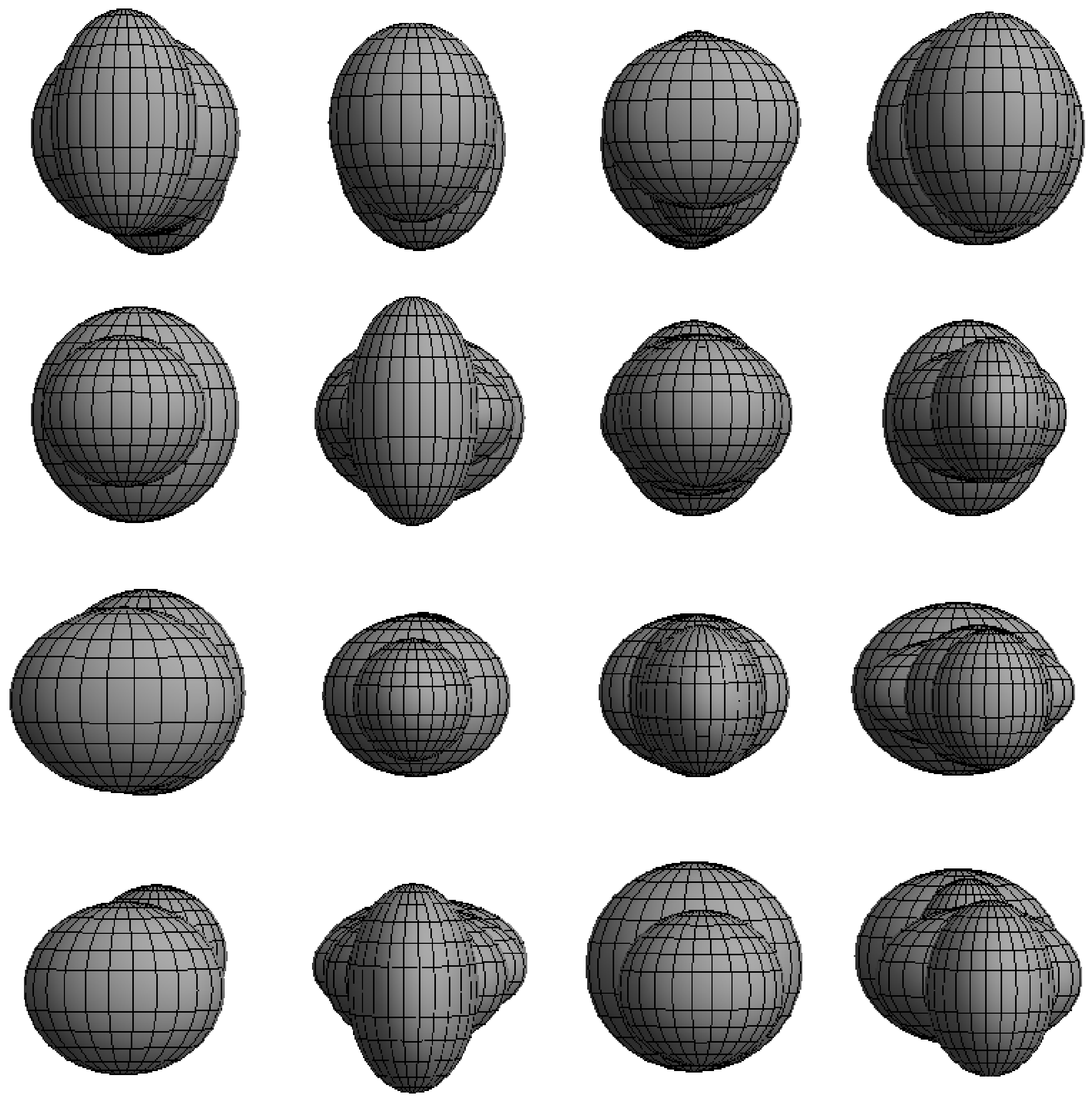


Acoustic scattering on the exterior domain. Each bowl is about 5λ.

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. *[2015, CAMWA, Hao/M./Young]*

Fast direct solvers and high order methods: Boundary integral equations

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.)

[2015, BIT, with Bremer/Gillman/Martinsson.]

Fast direct solvers and high order methods: Boundary integral equations

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

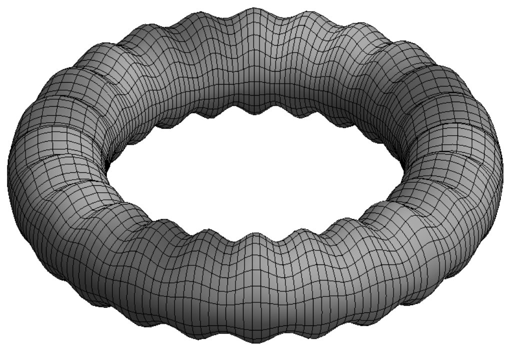
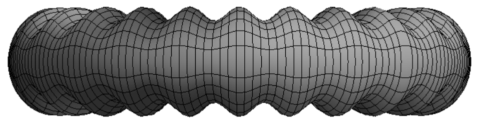
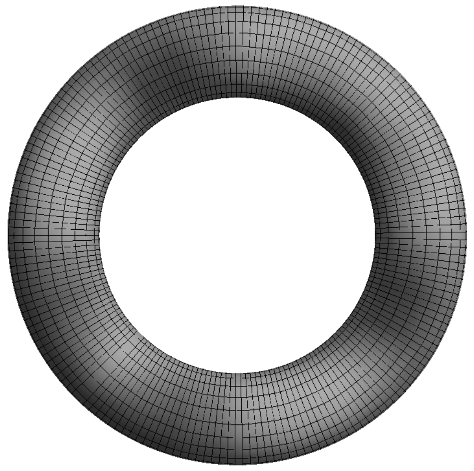
Increasing the accuracy is possible, but comes at a cost.

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

[2015, BIT, with Bremer/Gillman/Martinsson.]

Fast direct solvers and high order methods: Boundary integral equations

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.)

[2015, BIT, with Bremer/Gillman/Martinsson.]

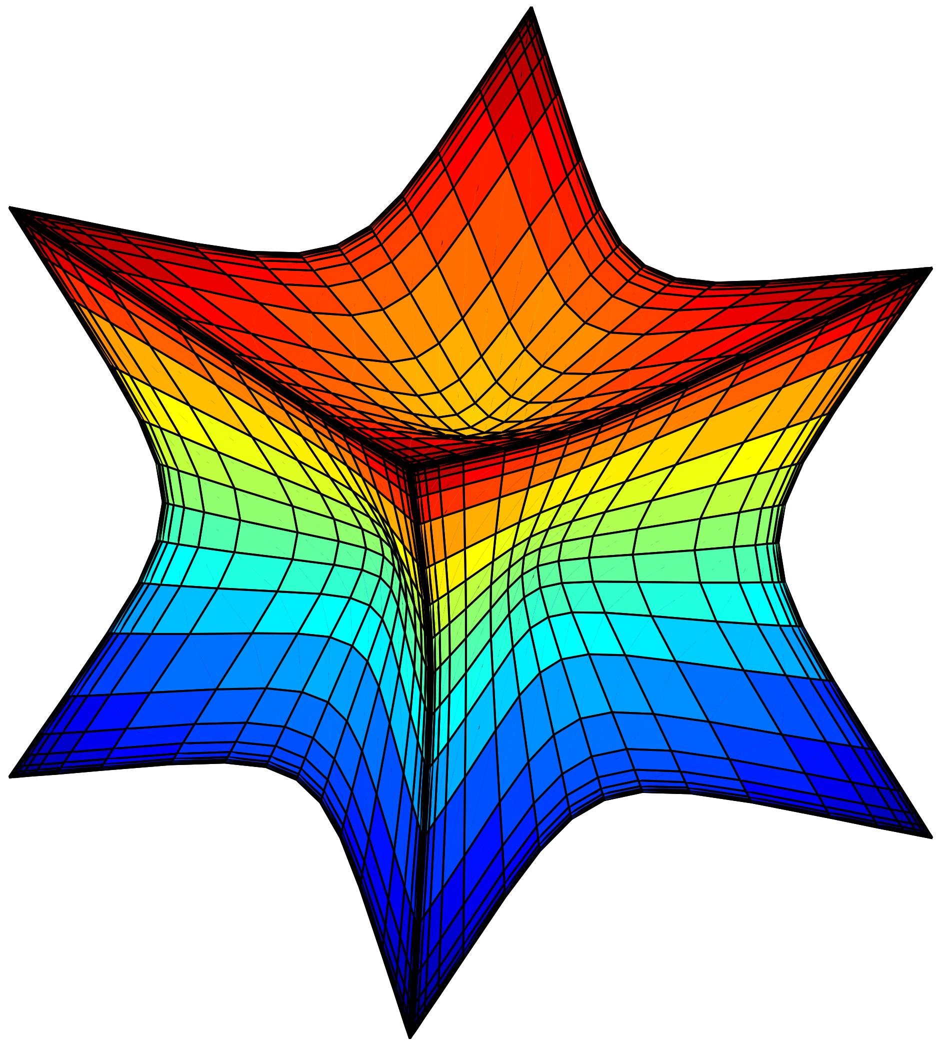
Fast direct solvers and high order methods: Boundary integral equations

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.

[2015, BIT, with Bremer/Gillman/Martinsson.]

Fast direct solvers and high order methods: "FEM-BEM coupling"

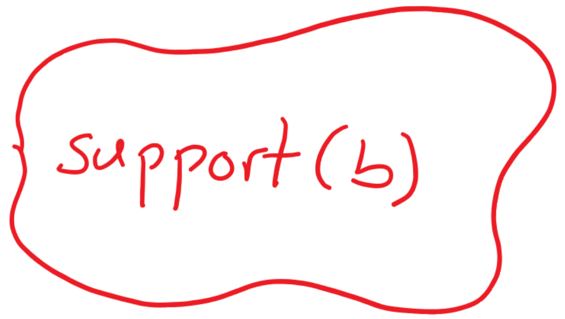
Consider the free space acoustic scattering problem

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

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."

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



Joint work with A. Barnett and A. Gillman.

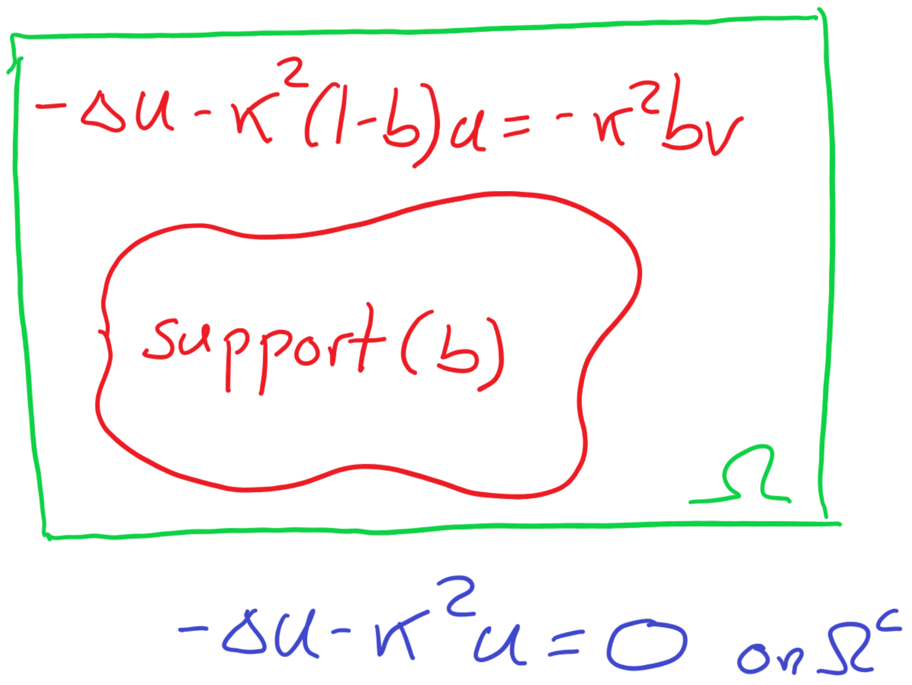
Fast direct solvers and high order methods: "FEM-BEM coupling"

Consider the free space acoustic scattering problem

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

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."


$-\Delta u - \kappa^2(1-b)u = -\kappa^2 bv$
support(b)
Ω
$-\Delta u - \kappa^2 u = 0$ on Ω^c

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

Joint work with A. Barnett and A. Gillman.

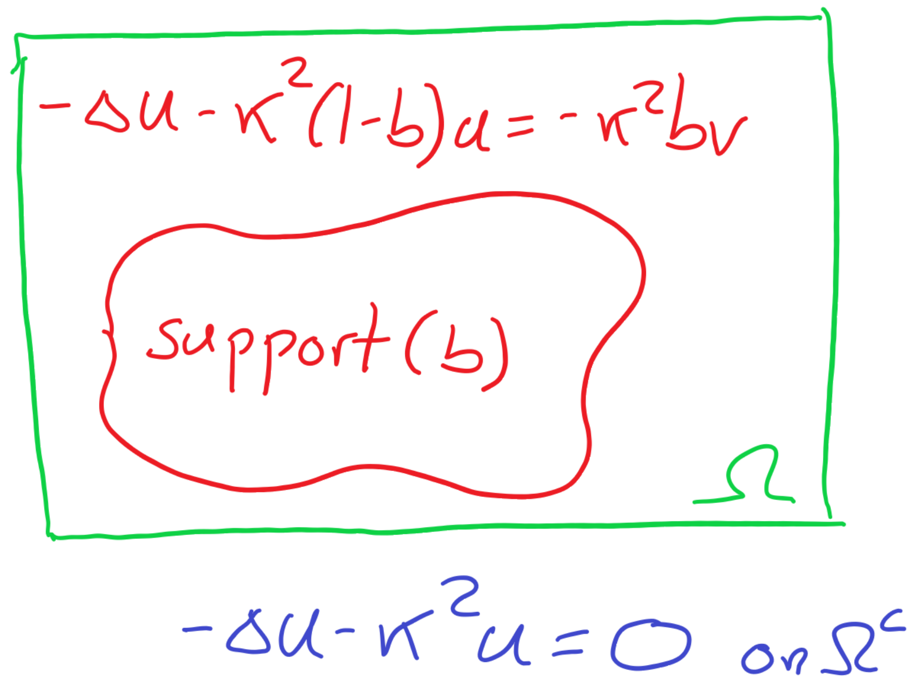
Fast direct solvers and high order methods: "FEM-BEM coupling"

Consider the free space acoustic scattering problem

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

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."


$-\Delta u - \kappa^2(1-b)u = -\kappa^2 bv$
support(b)
Ω
$-\Delta u - \kappa^2 u = 0$ on Ω^c

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

On Ω:

- Variable coefficient PDE.

On Ω^c:

- Constant coefficient PDE.

Joint work with A. Barnett and A. Gillman.

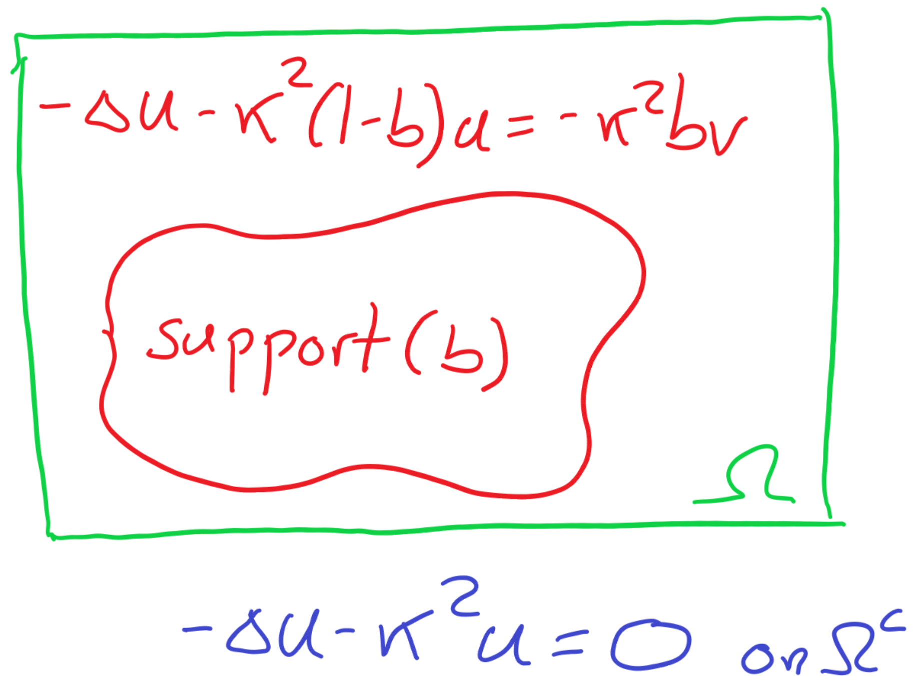
Fast direct solvers and high order methods: "FEM-BEM coupling"

Consider the free space acoustic scattering problem

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

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."



$-\Delta u - \kappa^2(1-b)u = -\kappa^2 bv$
support(b)
Ω
$-\Delta u - \kappa^2 u = 0$ on Ω^c

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

On Ω:

- Variable coefficient PDE.
- Use HPS.

On Ω^{c}:

- Constant coefficient PDE.
- Use BIE.

Joint work with A. Barnett and A. Gillman.

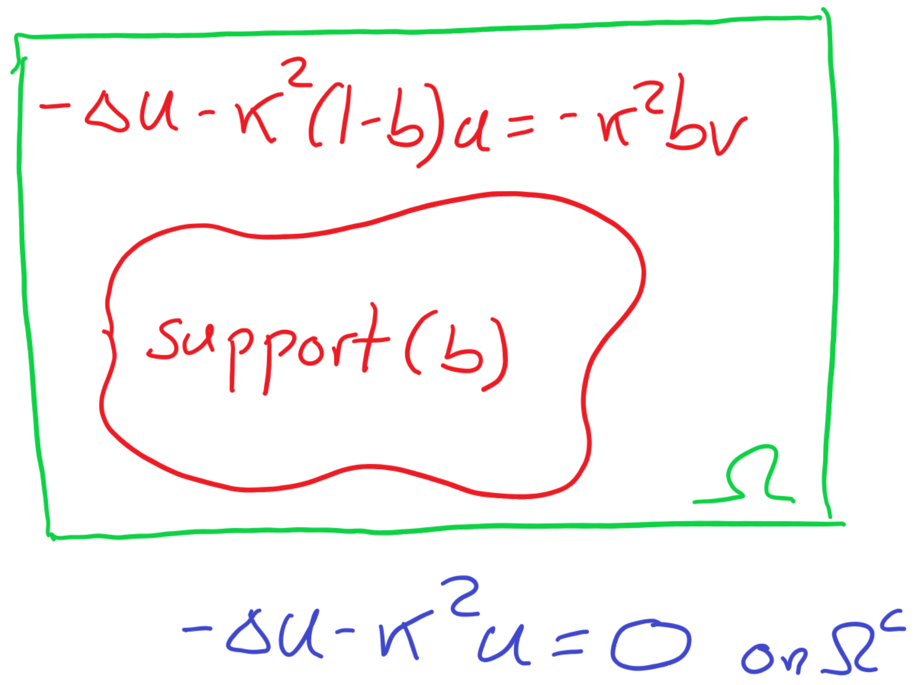
Fast direct solvers and high order methods: "FEM-BEM coupling"

Consider the free space acoustic scattering problem

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

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."



$-\Delta u - \kappa^2(1-b)u = -\kappa^2 bv$
support(b)
Ω
$-\Delta u - \kappa^2 u = 0$ on Ω^c

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

On Ω:	On Ω^c:
• Variable coefficient PDE.	• Constant coefficient PDE.
• Use HPS.	• Use BIE.
• Build DtN for $\partial\Omega$.	• Build DtN for $\partial\Omega^c$.

Joint work with A. Barnett and A. Gillman.

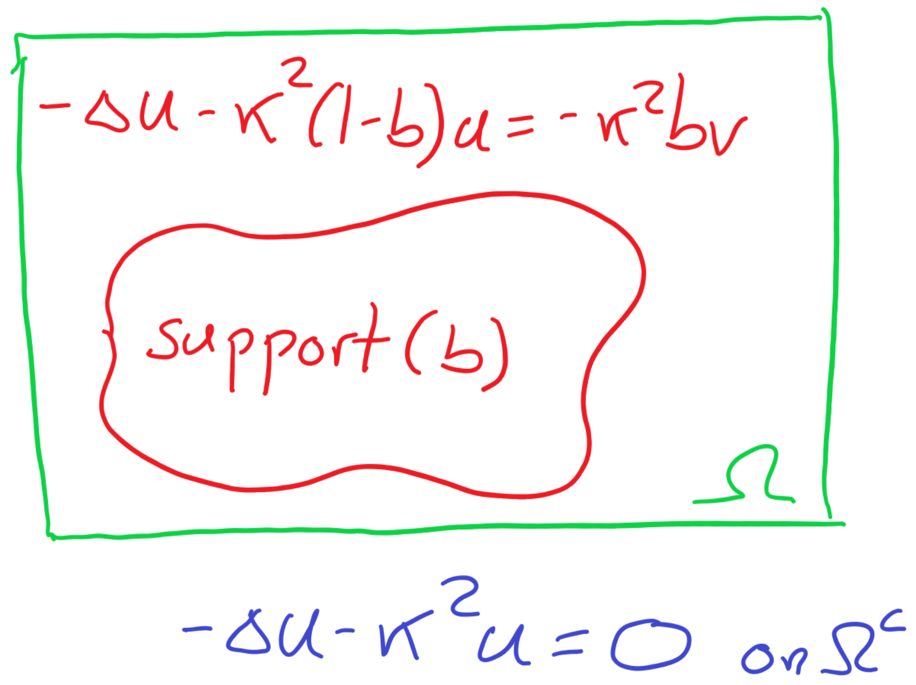
Fast direct solvers and high order methods: "FEM-BEM coupling"

Consider the free space acoustic scattering problem

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

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)
Ω
$-\Delta u - \kappa^2 u = 0$ on Ω^c

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

On Ω:	On Ω^c:
• Variable coefficient PDE.	• Constant coefficient PDE.
• Use HPS.	• Use BIE.
• Build DtN for $\partial\Omega$.	• Build DtN for $\partial\Omega^c$.

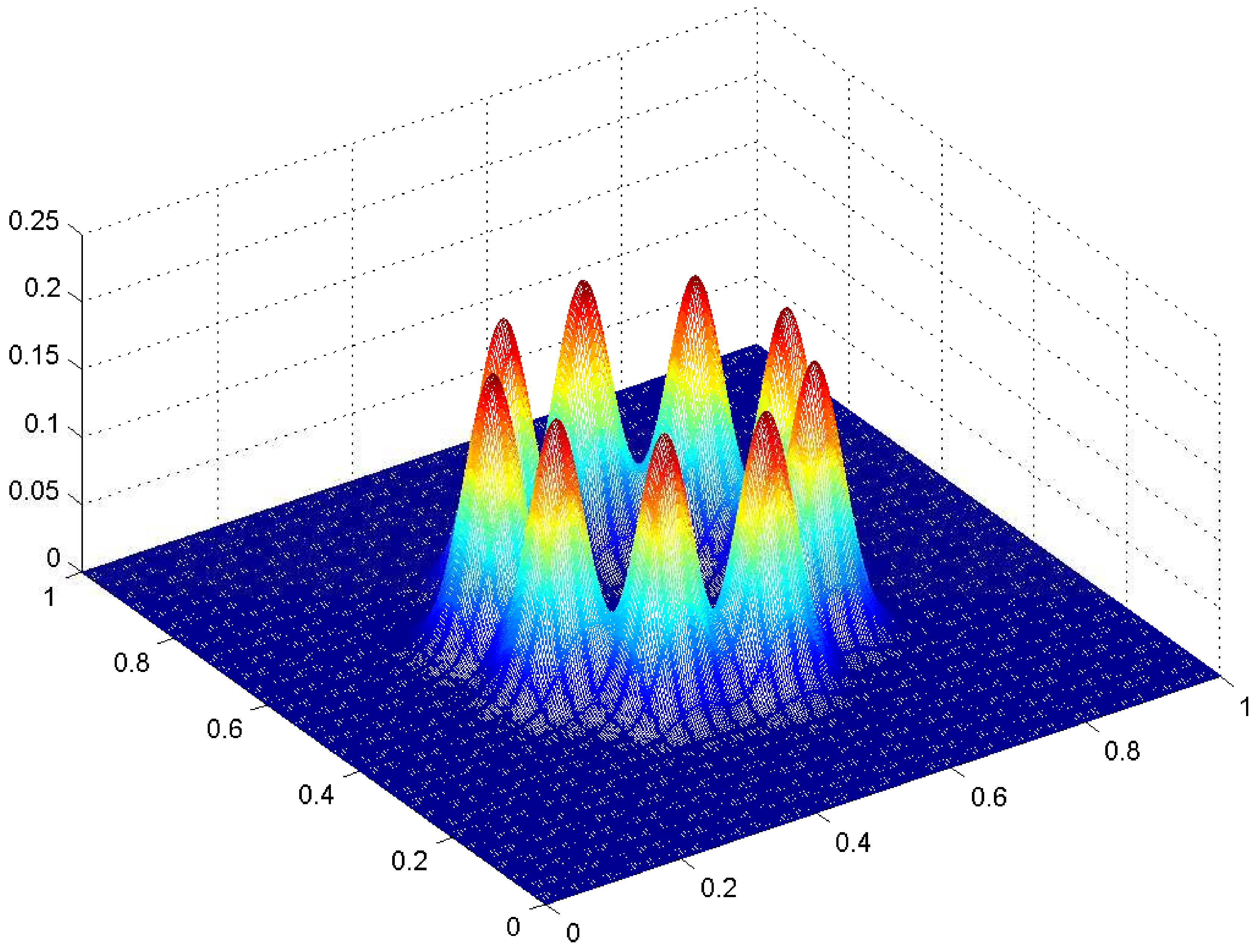
• *Merge using fast operator algebra!*

Joint work with A. Barnett and A. Gillman.

Fast direct solvers and high order methods: "FEM-BEM coupling"

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

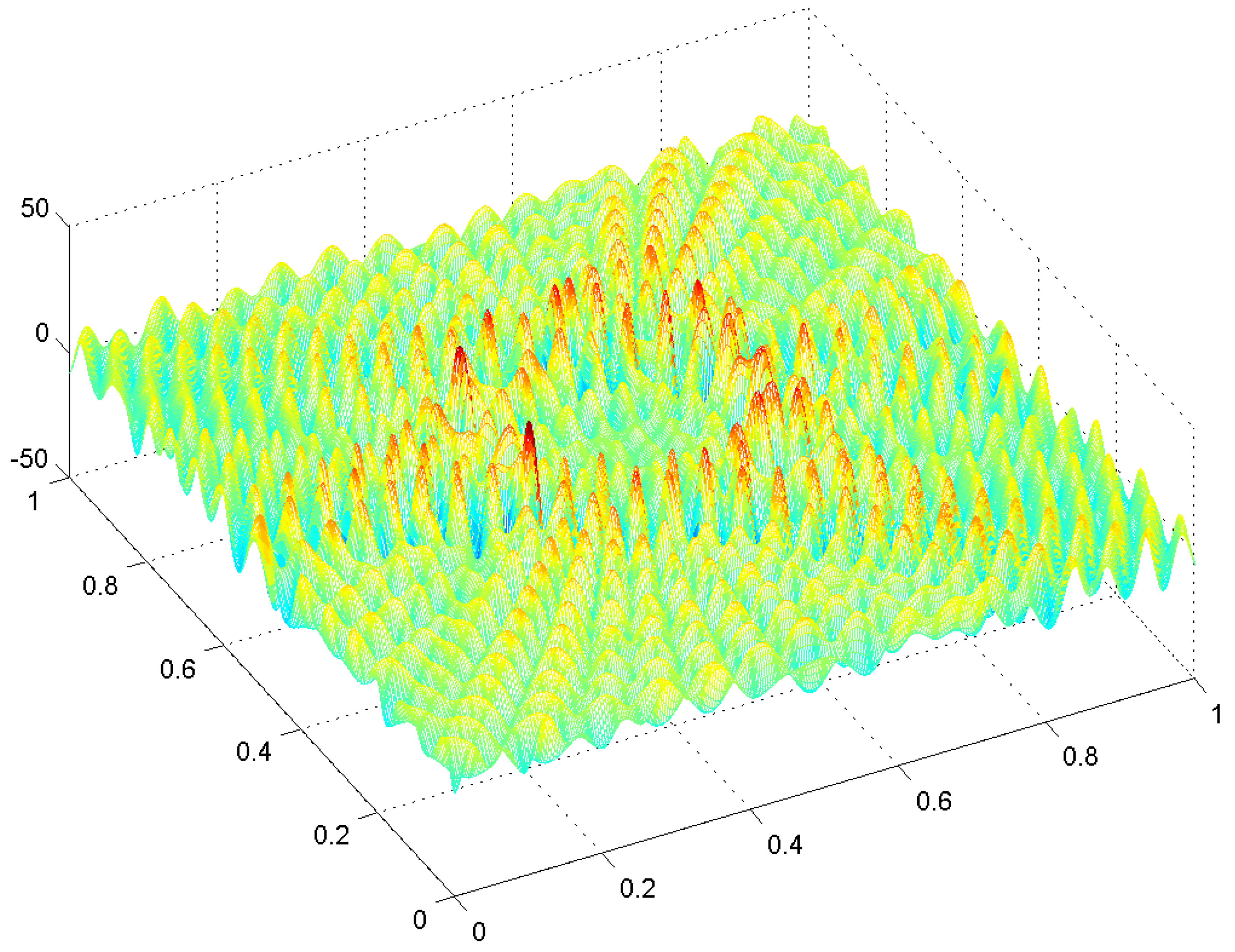
The scattering potential b



Fast direct solvers and high order methods: "FEM-BEM coupling"

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

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

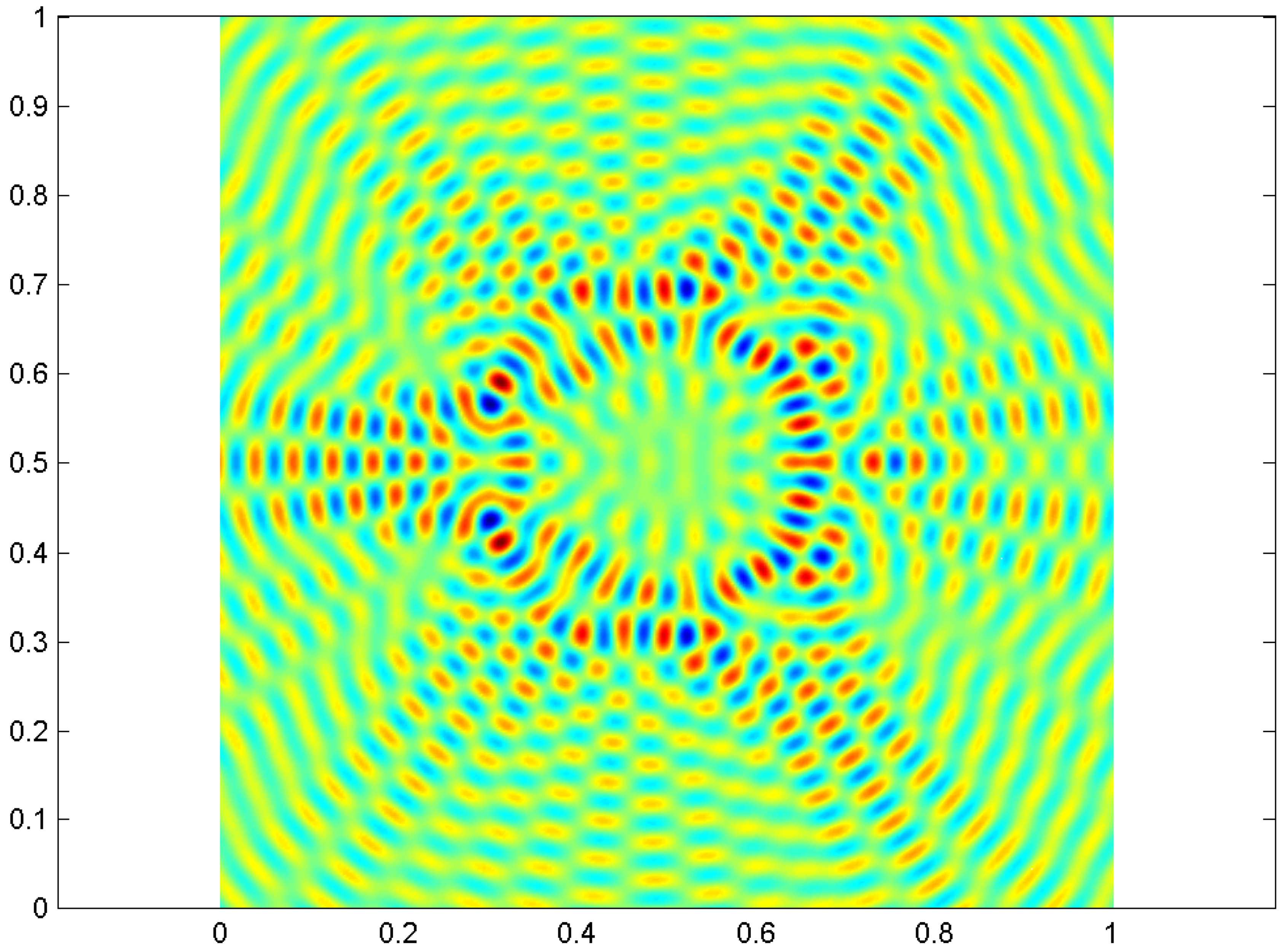


$N = 231\,361$ $T_{\rm build} = 7.2$ sec $T_{\rm solve} = 0.06$ sec $E \approx 10^{-7}$ (estimated)

Fast direct solvers and high order methods: “FEM-BEM coupling”

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

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

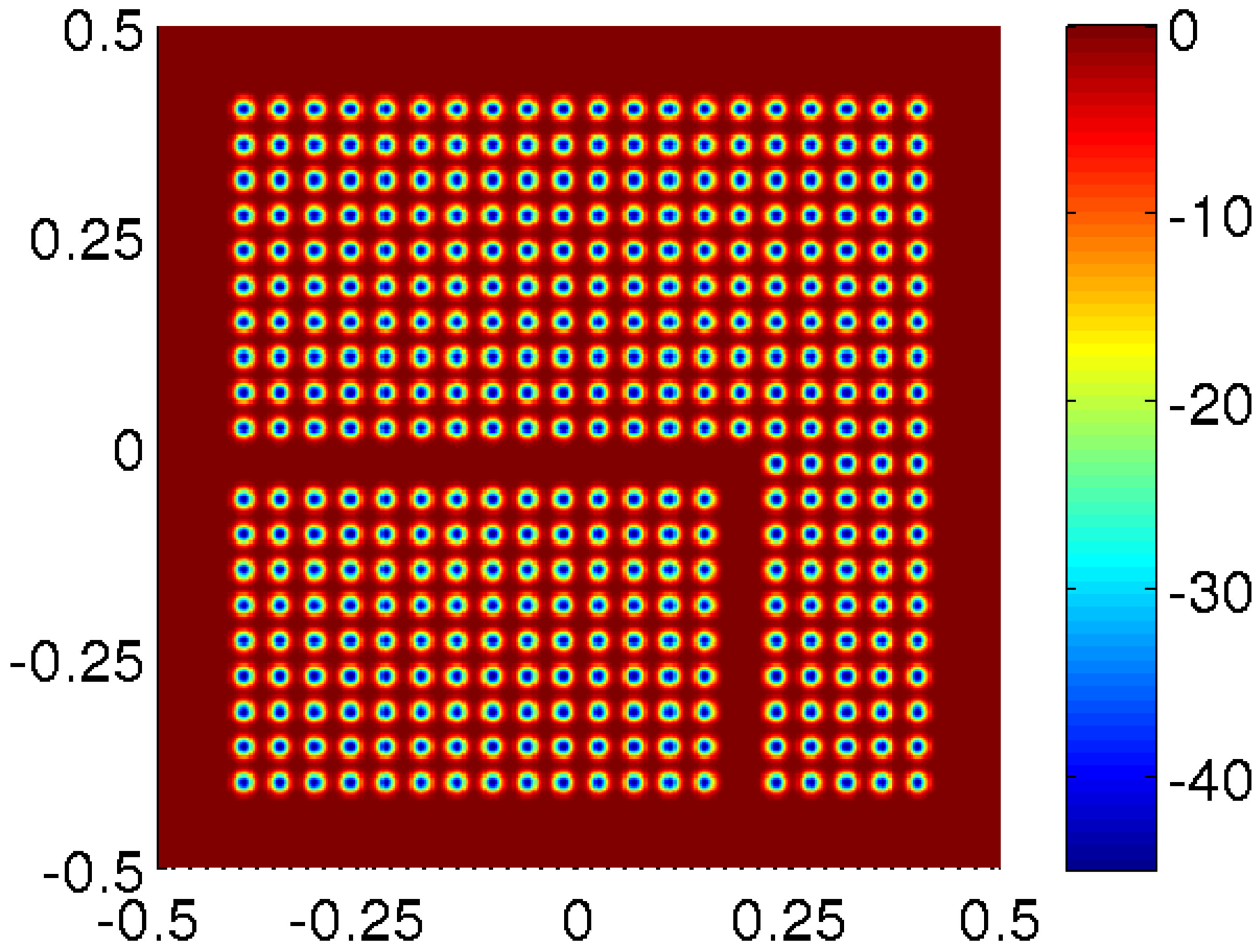


$N = 231\,361$ $T_{\rm build} = 7.2$ sec $T_{\rm solve} = 0.06$ sec $E \approx 10^{-7}$ (estimated)

Fast direct solvers and high order methods: "FEM-BEM coupling"

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

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

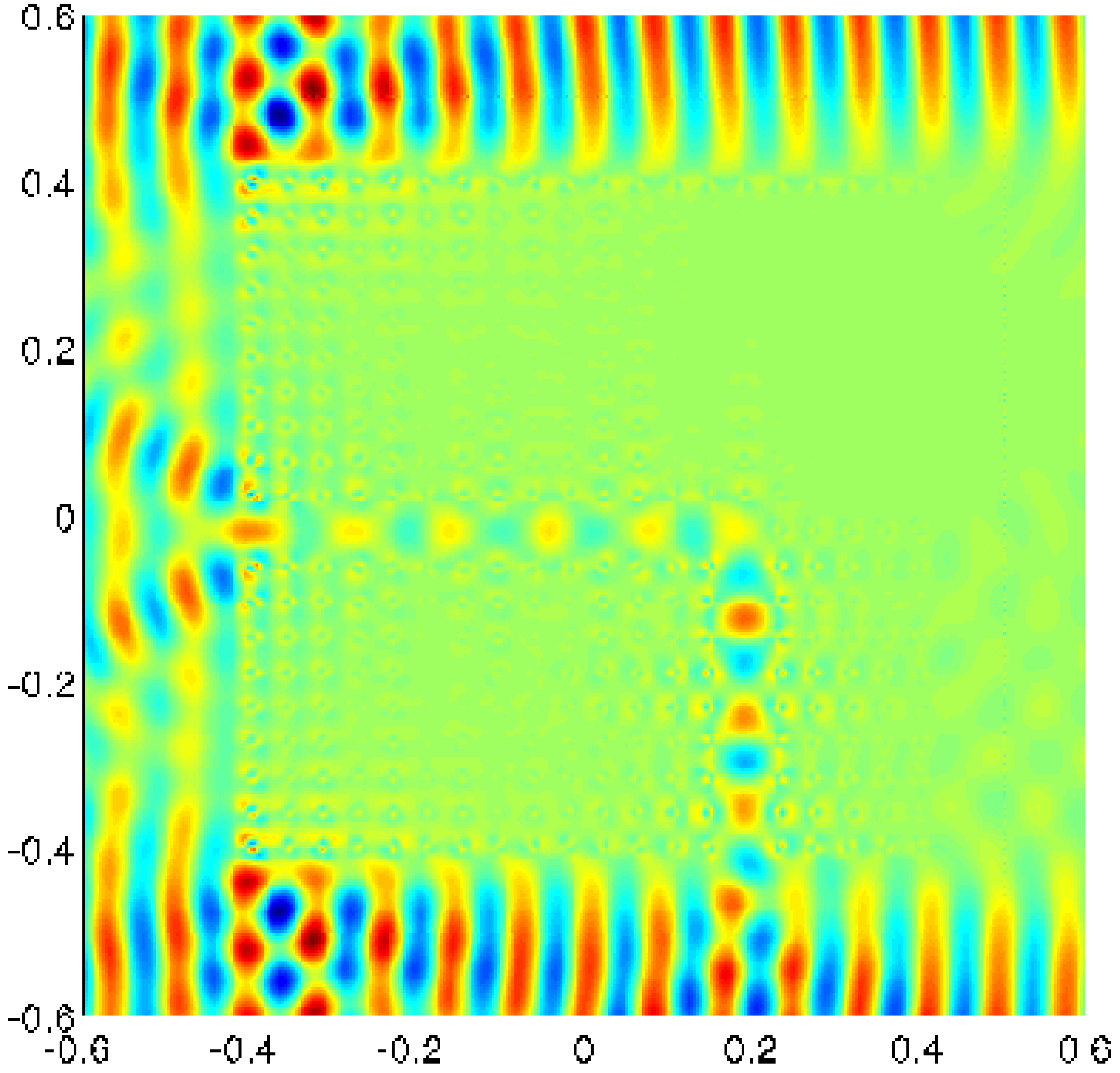


$N = 231\,361$ $\quad T_{\rm build} = 7.2$ sec $\quad T_{\rm solve} = 0.06$ sec $\quad E \approx 10^{-6}$ (estimated)

Fast direct solvers and high order methods: "FEM-BEM coupling"

$$\begin{cases} -\Delta u_{\rm out}(\boldsymbol{x}) - \kappa^2 (1 - b(\boldsymbol{x}))\, u_{\rm out}(\boldsymbol{x}) = -\kappa^2\, b(\boldsymbol{x})\, u_{\rm in}(\boldsymbol{x}) \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|} u_{\rm out}(\boldsymbol{x}) - i\kappa\, u_{\rm out}(\boldsymbol{x})\big) = 0 \end{cases}$$

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



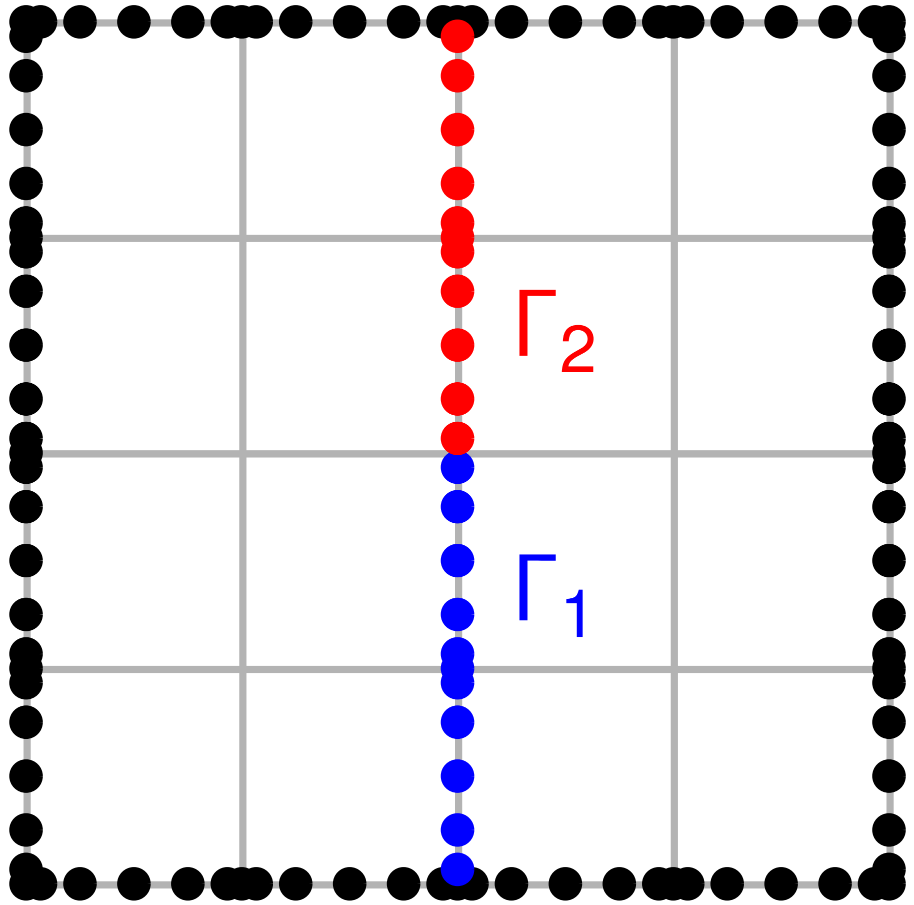
Fast direct solvers and high order methods: rank deficiencies in HPS

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?*

Fast direct solvers and high order methods: rank deficiencies in HPS

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?*

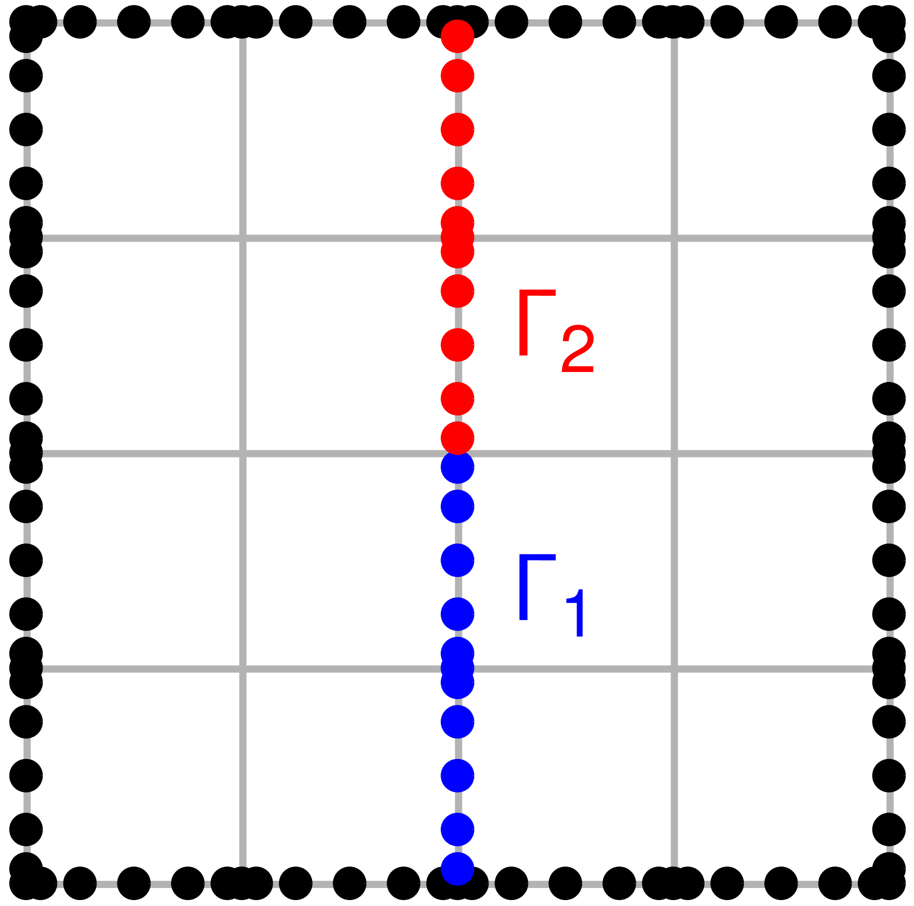
Let **T** denote the restriction of the DtN matrix mapping Dirichlet data on Γ_1 to Neumann data on Γ_2 for a $1\,089 \times 1\,089$ grid. Then **T** is of size 512×512.



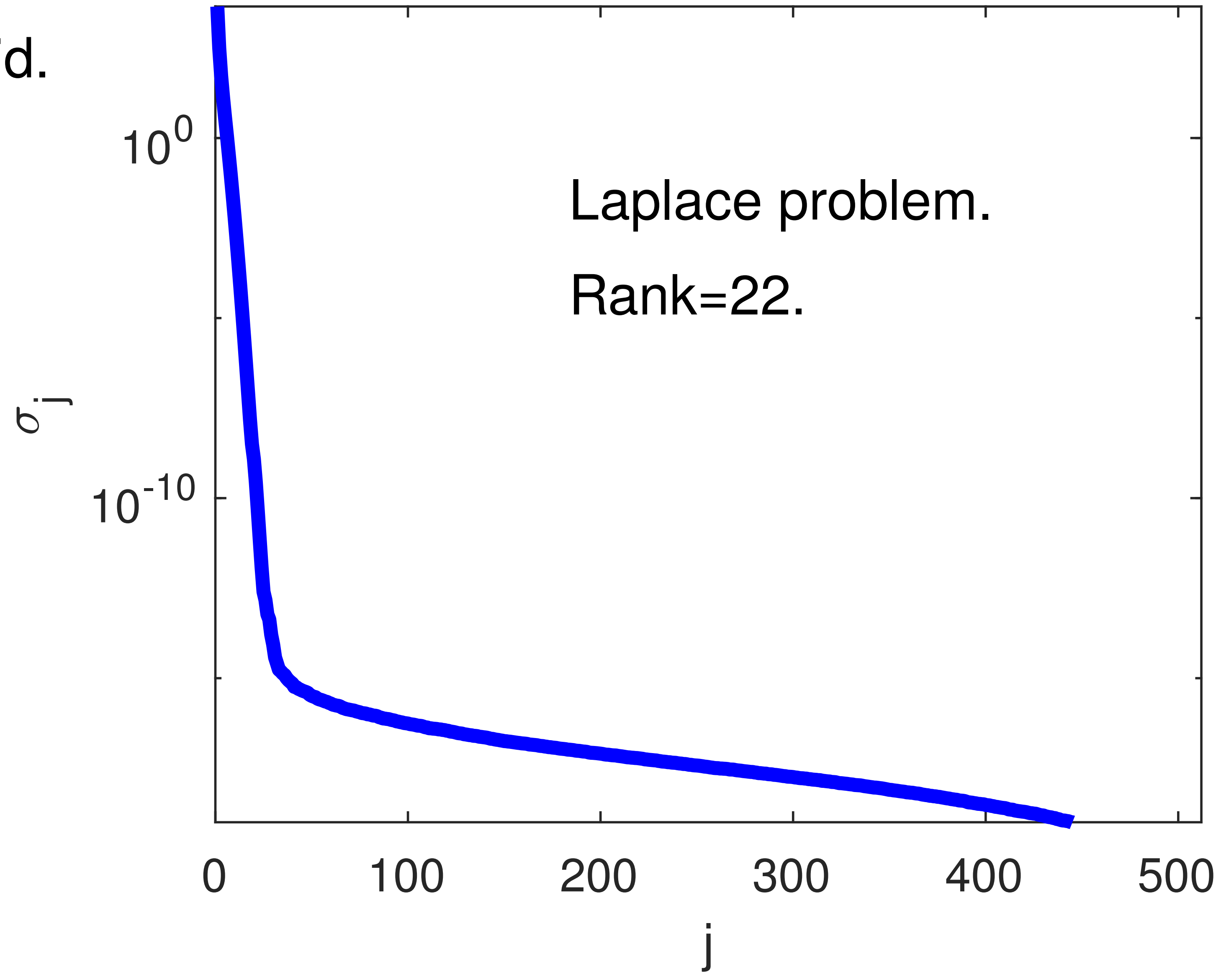
Fast direct solvers and high order methods: rank deficiencies in HPS

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 **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 **T** is of size 512×512.



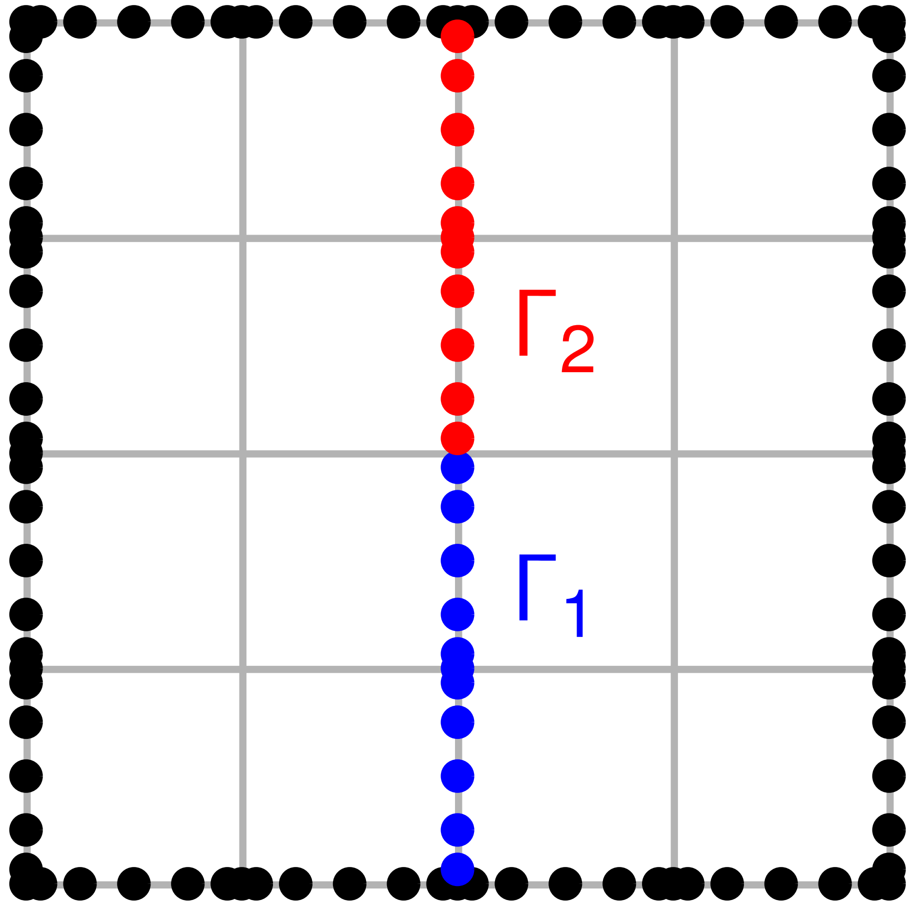
Singular values of **T***.*



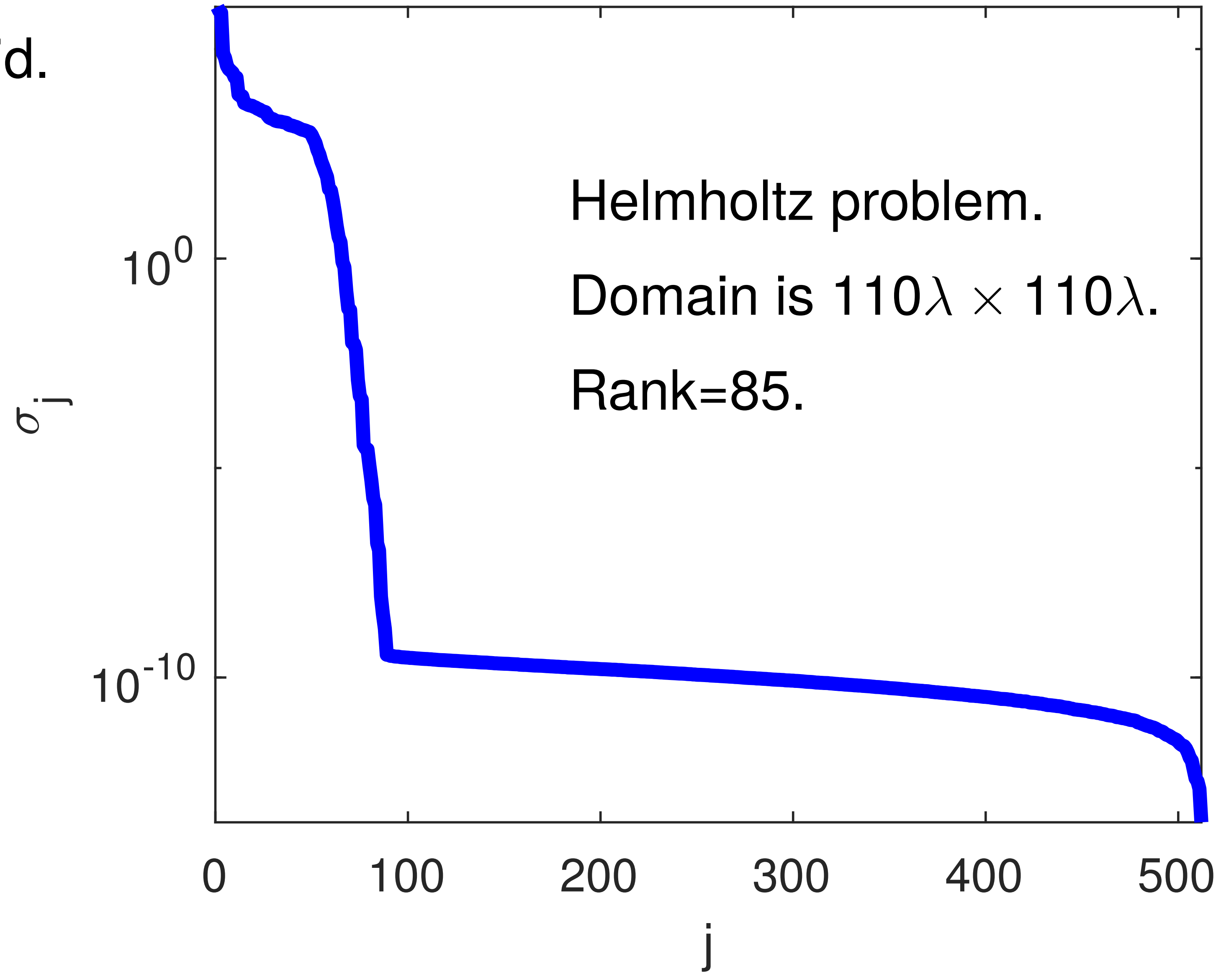
Fast direct solvers and high order methods: rank deficiencies in HPS

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 **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 **T** is of size 512×512.



Singular values of **T**.



Helmholtz problem.

Domain is $110\lambda \times 110\lambda$.

Rank=85.

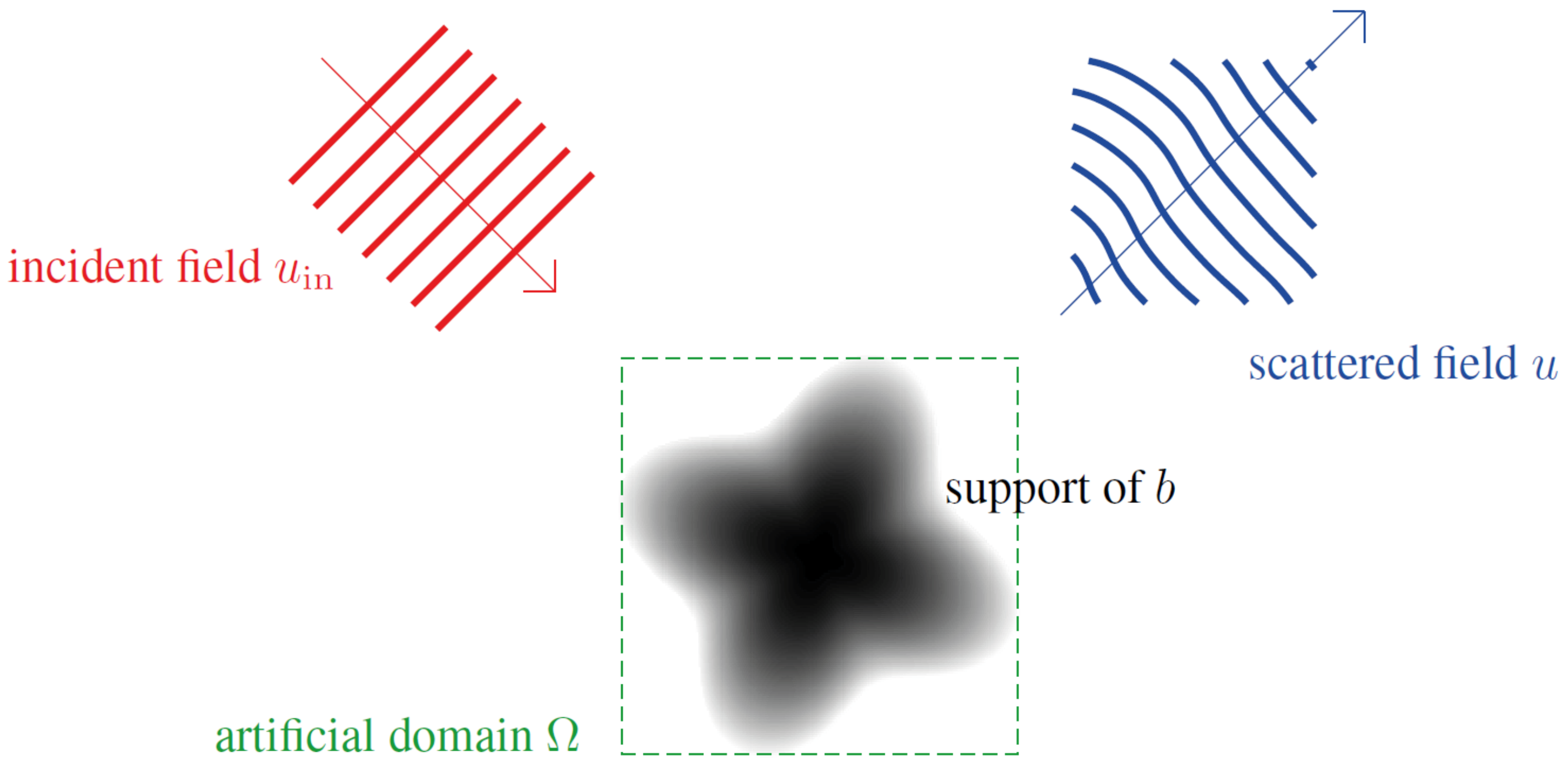
Fast direct solvers and high order methods: Lippmann-Schwinger

Consider the free space acoustic scattering problem

$$(6)\qquad \begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2\,(1 - b(\boldsymbol{x}))\,u(\boldsymbol{x}) = -\,\kappa^2\,b(\boldsymbol{x})\,v(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^2 \\ \lim_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\,\big(\partial_{|\boldsymbol{x}|}u(\boldsymbol{x}) - i\kappa\,u(\boldsymbol{x})\big) = 0, & \end{cases}$$

where

- b is a smooth scattering potential with compact support in a domain Ω, where
- v is a given "incoming potential" and where
- u is the sought "outgoing potential."



In the figure, $v = u_{\rm in}$.

Fast direct solvers and high order methods: Lippmann-Schwinger

Consider the free space acoustic scattering problem

$$(6)\qquad \begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2\,(1 - b(\boldsymbol{x}))\,u(\boldsymbol{x}) = -\,\kappa^2\,b(\boldsymbol{x})\,v(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^2 \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\big(\partial_{|\boldsymbol{x}|}u(\boldsymbol{x}) - i\kappa\,u(\boldsymbol{x})\big) = 0, & \end{cases}$$

where

- b is a smooth scattering potential with compact support in a domain Ω, where
- v is a given "incoming potential" and where
- u is the sought "outgoing potential."

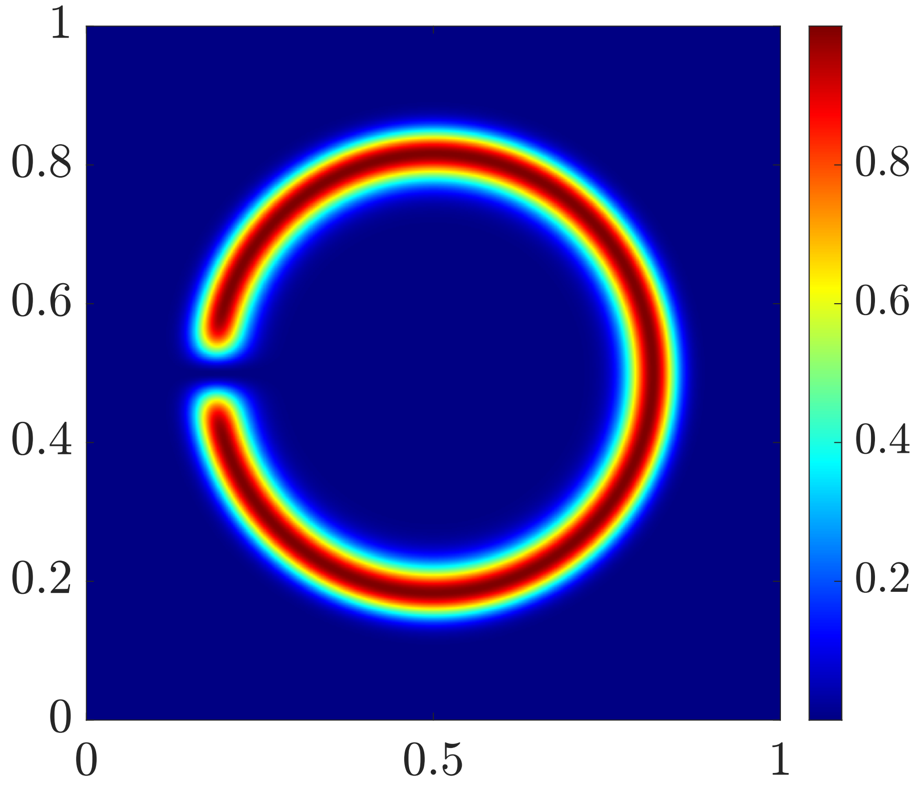
Let us now use an integral equation formulation. It is well known that (6) has an alternative formulation in the *Lippmann-Schwinger integral equation*

$$(7)\qquad \sigma(\boldsymbol{x}) + \kappa^2 b(\boldsymbol{x}) \int_\Omega \phi_\kappa(\boldsymbol{x} - \boldsymbol{y})\sigma(\boldsymbol{y})\,d\boldsymbol{y} = -\kappa^2 b(\boldsymbol{x})v(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega,$$

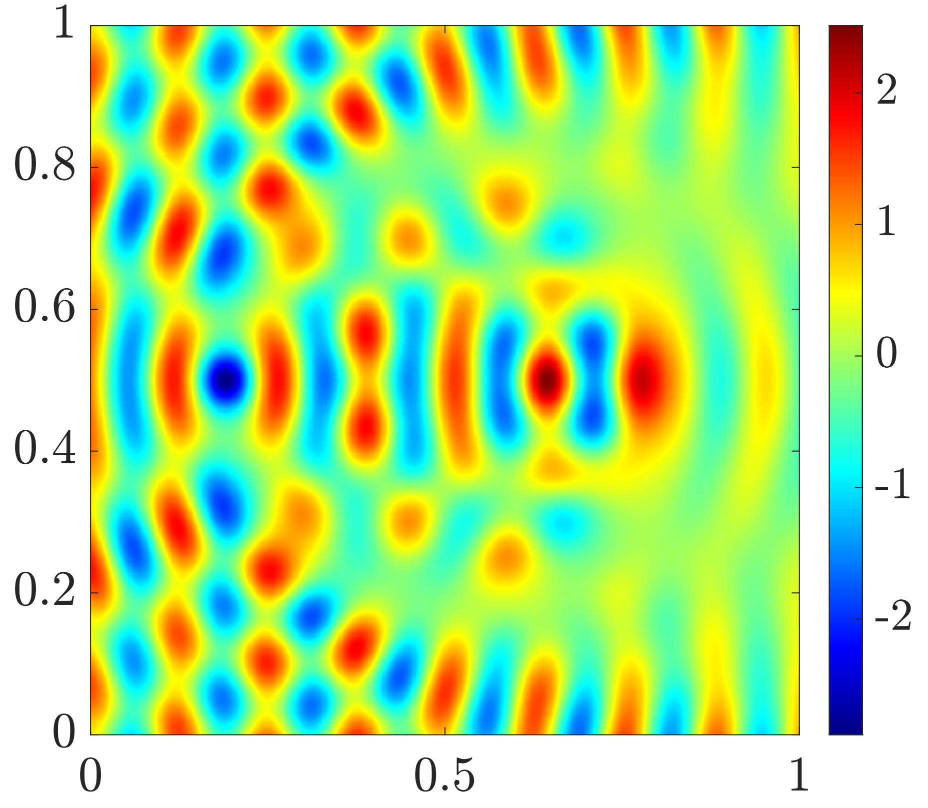
where ϕ_κ is the free space fundamental solution of the Helmholtz equation.

We discretize (7) using the trapezoidal rule on a uniform grid, with Duan-Rokhlin quadrature corrections of order 10.

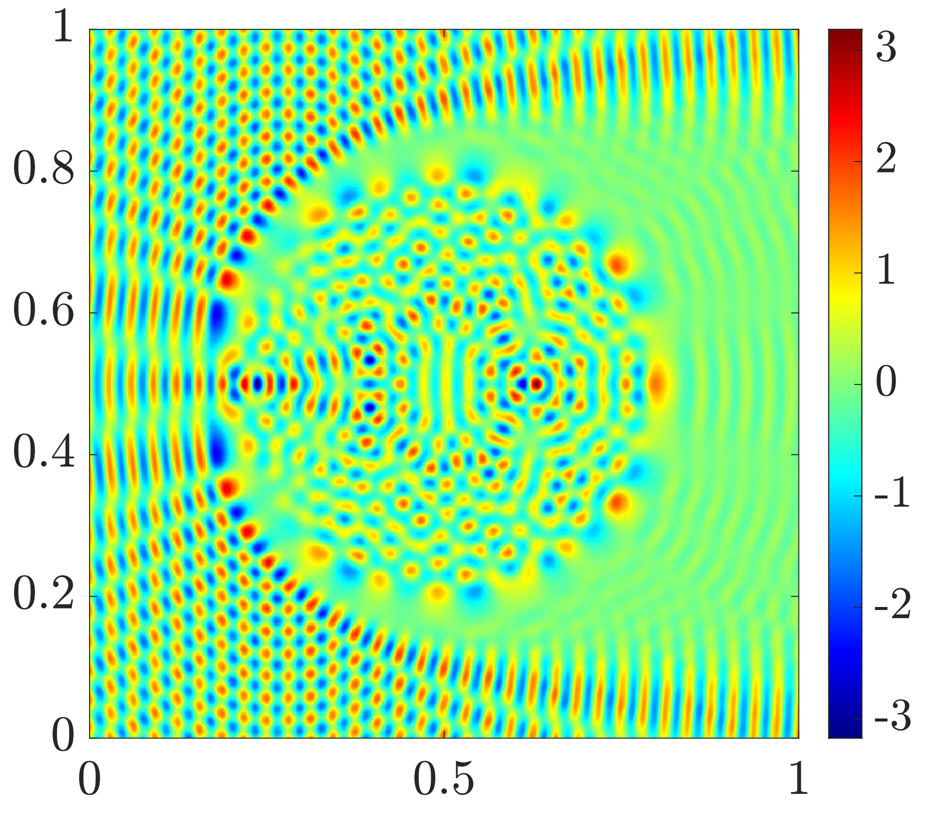
Fast direct solvers and high order methods: Lippmann-Schwinger



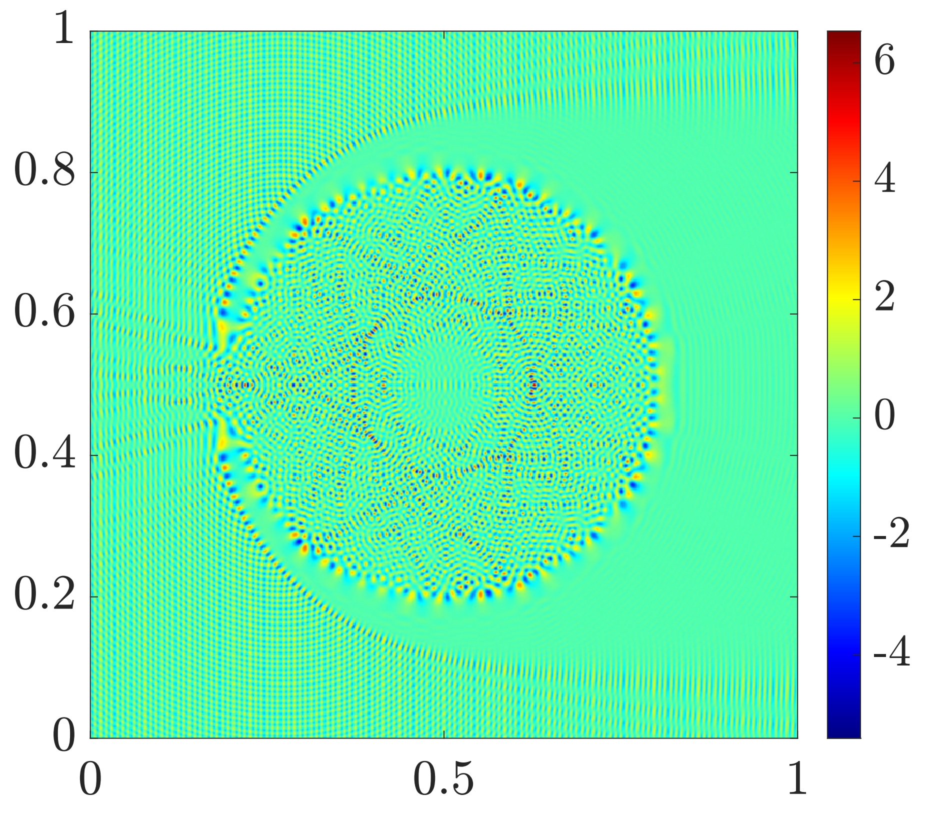
The scattering potential



$\kappa = 50$



$\kappa = 201$



$\kappa = 804$

Fast direct solvers and high order methods: Lippmann-Schwinger

Fixed 10 points per wavelength.

Direct solver is run at accuracy 10^{-3} and used as a preconditioner.

Weak admissibility is used.

N	κ	T_{build}	T_{inv}	T_{gmres}	mem	iter	res
6400	50.27	0.23	0.24	0.20	0.04	4	6.97e-11
25600	100.53	0.65	0.99	0.62	0.21	5	6.16e-12
102400	201.06	2.26	4.36	2.49	1.01	6	1.04e-12
409600	402.12	14.91	20.06	9.78	4.67	6	3.23e-11
1638400	804.25	99.01	91.37	56.13	21.16	9	8.12e-12
6553600	1608.50	430.60	398.88	330.91	94.63	13	3.93e-11
26214400	3216.99	3102.09	2024.16	2698.53	418.37	22	3.30e-11

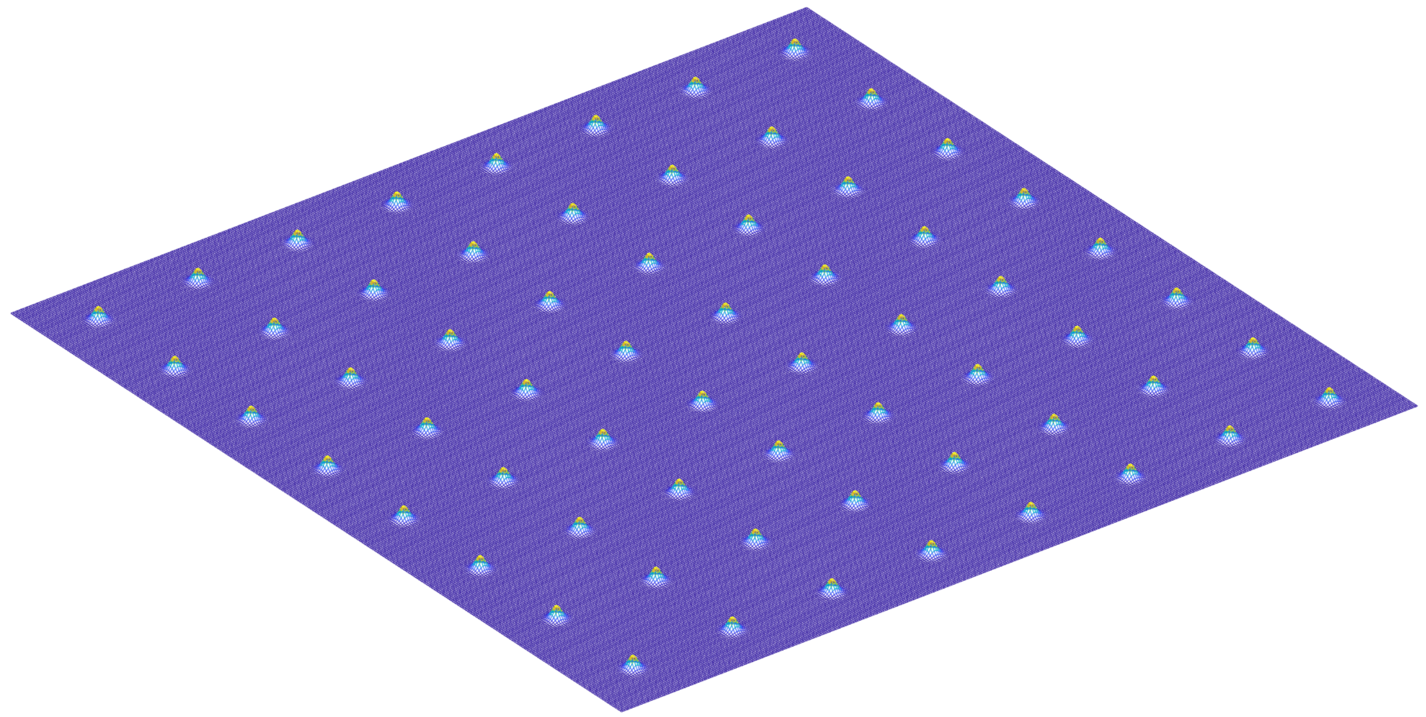
The largest experiment is over 500λ in diameter: Less than 3h total run time.

Hardware: Workstation with dual Intel Xeon Gold 6254 (18 cores at 3.1GHz base frequency).

Joint with Abi Gopal, arxiv #2007.12718

Fast direct solvers and high order methods: rough surface scattering

Consider acoustic scattering from an infinite half-plane with 8×8 bumps:



Each bump is 0.5λ tall and 2λ wide. Total domain is $44\lambda \times 44\lambda$.

The problem is formulated as a BIE, and is discretized using a “Zeta-corrected” quadrature rule. A direct solver with weak admissibility and $O(N^{1.5})$ scaling is used.

26M points total, $T_{\rm build} = 1200s$, $T_{\rm solve} = 184s$

About three digits of accuracy in the computed solution. (Tentative!)

Work in progress - with Abi Gopal and Bowei Wu.

Active research area!

HSS-accelerated multifrontal solvers: J. Xia, M. de Hoop, X. Li, . . .

BLR-accelerated multifrontal solvers: P. Amestoy, C. Ashcraft, A. Buttari, J-Y. l'Excellent, T. Mary, . . .

Linear complexity recursive skeletonization: K. Ho, L. Ying.

Linear complexity inverse FMM / strong recursive skeletonization: S. Ambikasaran, E. Darve; V. Minden, K. Ho, A. Damle, L. Ying; M. O'Neil, D. Sushnikova, M. Rachh, L. Greengard; . . .

High frequency problems: Y. Liu, H. Guo, E. Michielssen; B. Engquist, L. Ying; S. Li, Y. Liu, P. Ghysels, L. Klaus; S. Börm, C. Börst; B. Bonev, J. Hesthaven; . . .

HPC and heterogenous computing environments: D. Keyes, H. Ltaief, G. Turkiyyah; G. Biros, C. Chen; . . .

High order spectral element methods: A. Townsend, D. Fortunato;

Apologies for omissions!

Outline of talk:

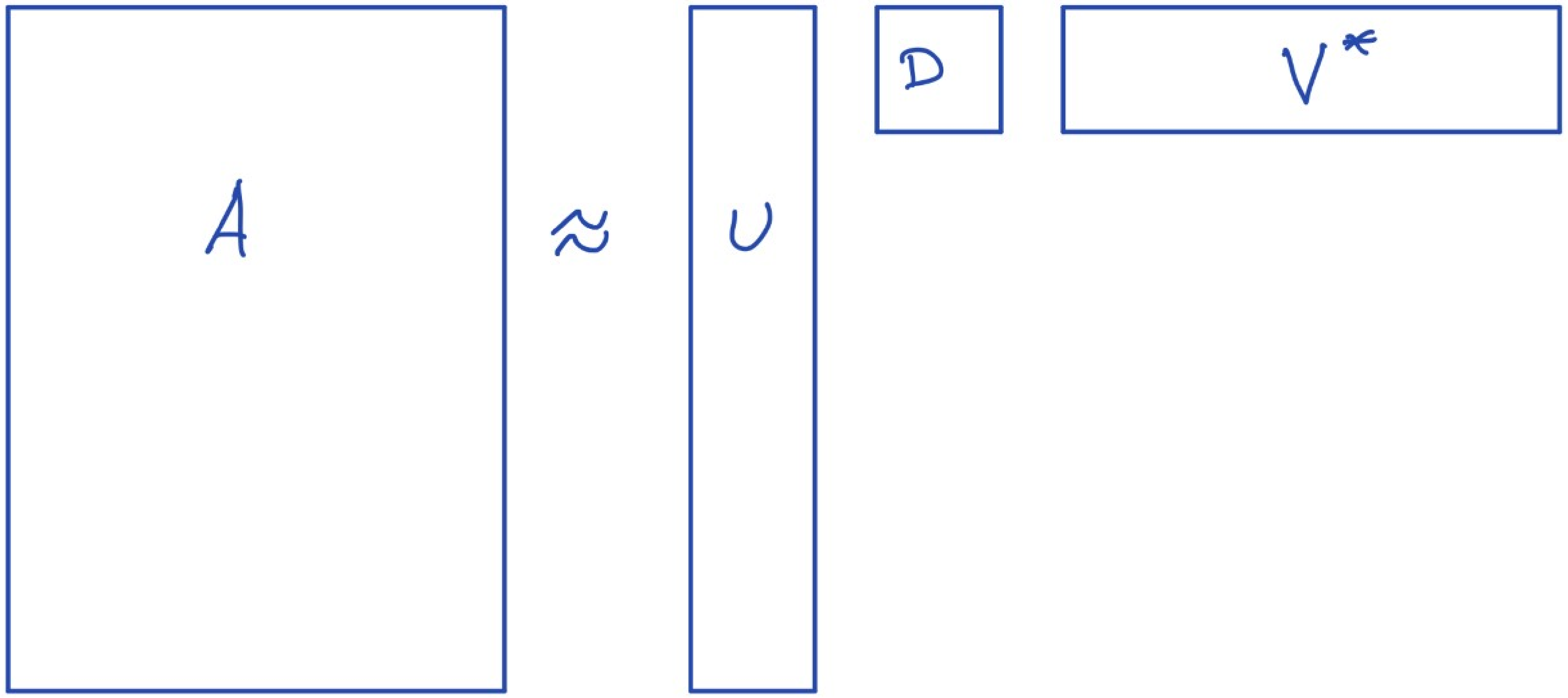
- Introduction: Problem formulation & solution operators. *[Done!]*
- Curse of dimensionality. *[Done!]*
- Interaction ranks — why are they small? How small are they? *[Done!]*
- (Versions of fast direct solvers — “strong” versus “weak” etc.) *[Skipped.]*
- High order discretizations and fast direct solvers. *[Done!]*
- Randomized low rank approximation.
- Randomized compression of rank structured matrices.

Randomized algorithms for low rank approximation:

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

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

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



Randomized algorithms for low rank approximation:

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

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

with **U** and **V** having orthonormal columns, and **D** diagonal.

Applications:

- Principal component analysis (fitting a hyperplane to data).
- Model reduction in analyzing physical systems.
- PageRank and other spectral methods in data analysis.
- Fast algorithms in scientific computing. (FMMs, Fast Direct Solvers, etc.)
- Diffusion geometry and manifold learning.
- Many, many more . . .

Classical deterministic methods: Gram-Schmidt (column pivoted QR), power/subspace iteration, Krylov techniques, . . .

Randomized algorithms for low rank approximation:

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

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

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

Solution:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\text{col}(\mathbf{Y}) = \text{col}(\mathbf{Q})$. `[Q, ~] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Why does it work? When $\mathbf{A}$ has exact rank k, the algorithm succeeds with probability 1.

Randomized algorithms for low rank approximation:

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

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

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

Solution:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\text{col}(\mathbf{Y}) = \text{col}(\mathbf{Q})$. `[Q, ~] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Why does it work? When $\mathbf{A}$ has exact rank k, the algorithm succeeds with probability 1. In the general case, it fails only if the columns of $\mathbf{G}$ manage to all be close to orthogonal to a dominant right singular vector. *Very* unlikely.

Randomized algorithms for low rank approximation:

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

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

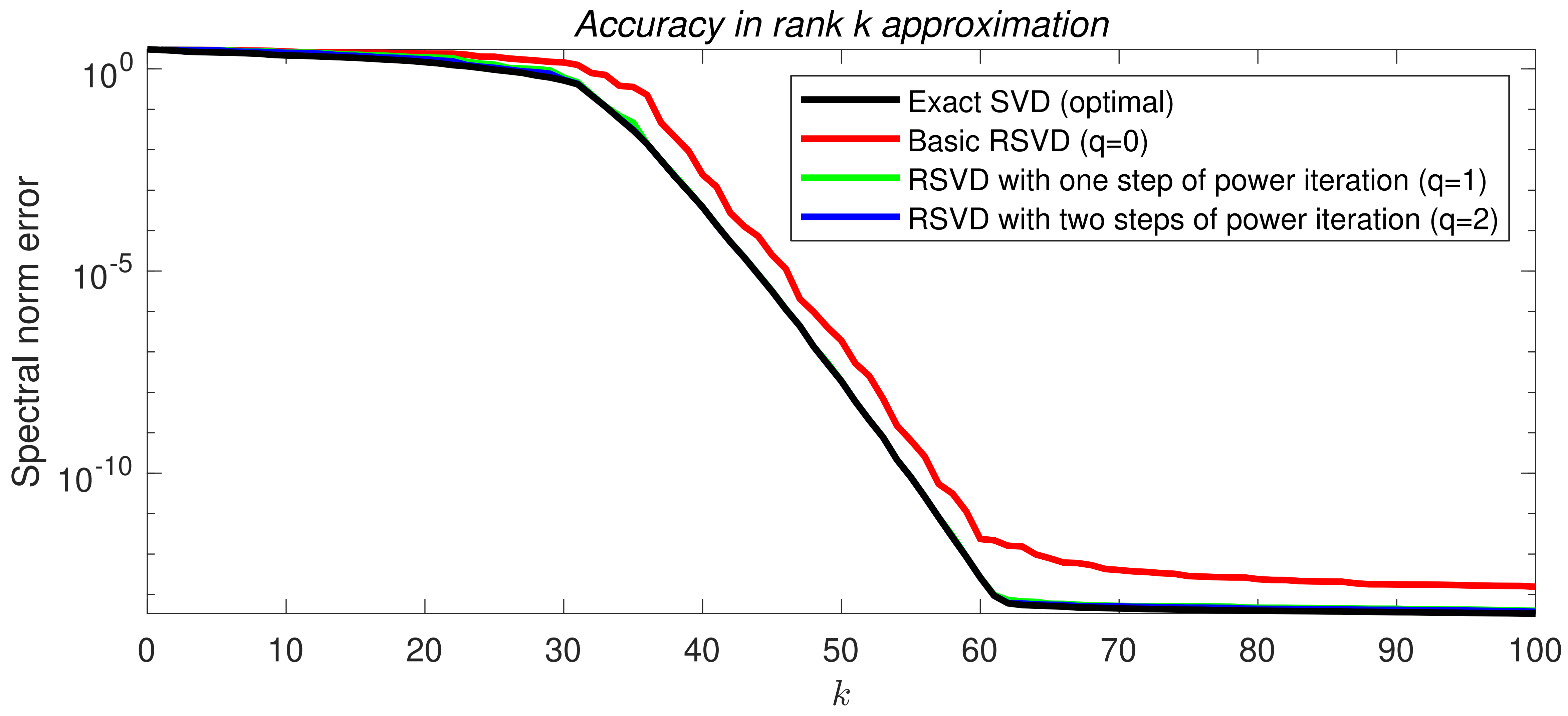
with $\mathbf{U}$ and $\mathbf{V}$ having orthonormal columns, and $\mathbf{D}$ diagonal.

Solution:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\text{col}(\mathbf{Y}) = \text{col}(\mathbf{Q})$. `[Q, ~] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Power iteration: When the singular values of $\mathbf{A}$ decay slowly, precision can be improved by replacing the formula $\mathbf{Y} = \mathbf{A}\mathbf{G}$ on line 2 by $\mathbf{Y} = \mathbf{A}(\mathbf{A}^*\mathbf{G})$, or $\mathbf{Y} = \mathbf{A}(\mathbf{A}^*(\mathbf{A}\mathbf{G}))$, or ...

Randomized low rank approximation:



The plot shows the errors from the randomized range finder. To be precise, we plot

$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

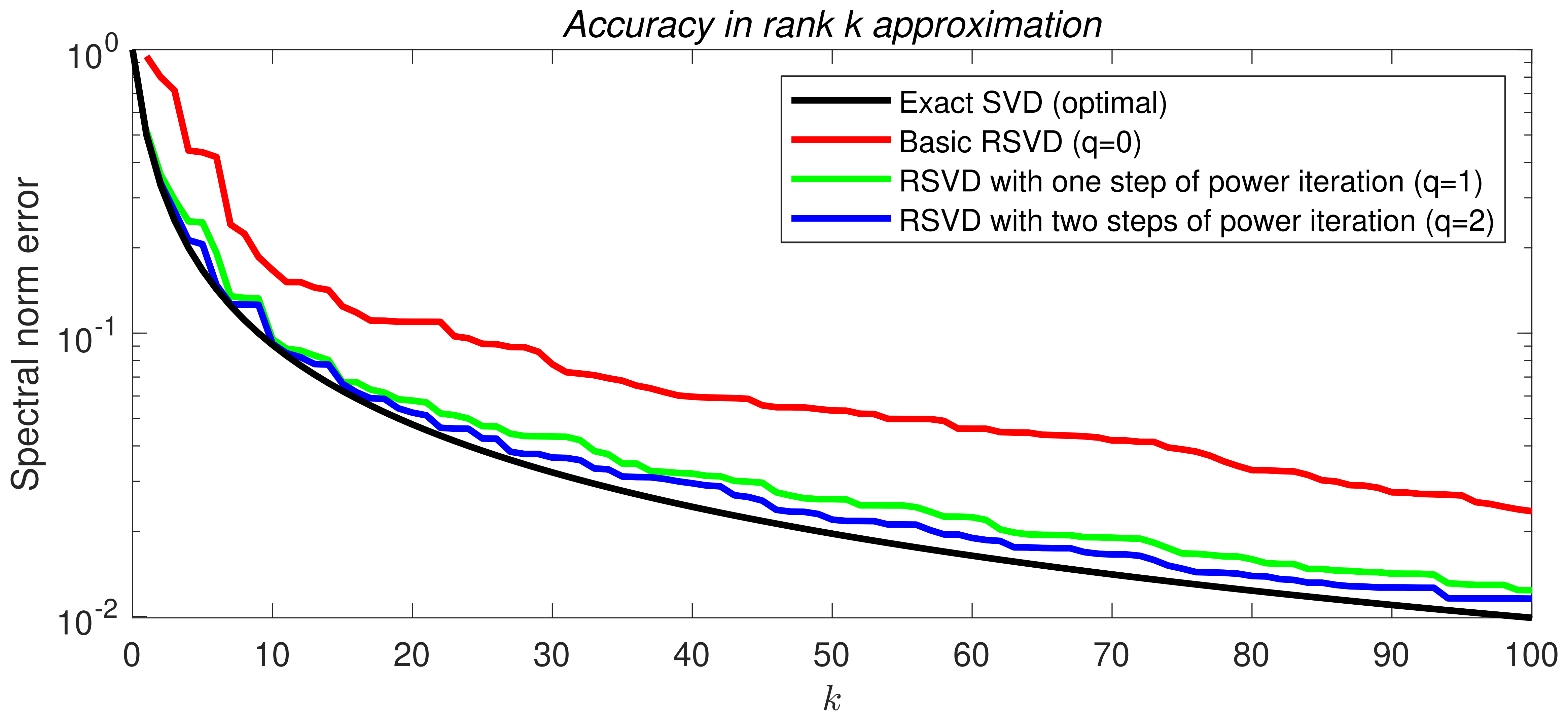
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

$$\mathbf{Y} = \left(\mathbf{A}\mathbf{A}^*\right)^q\mathbf{A}\mathbf{G},$$

and where $\mathbf{G}$ is a Gaussian random matrix.

The matrix $\mathbf{A}$ is an approximation to a scattering operator for a Helmholtz problem.

Randomized low rank approximation:



The plot shows the errors from the randomized range finder. To be precise, we plot

$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

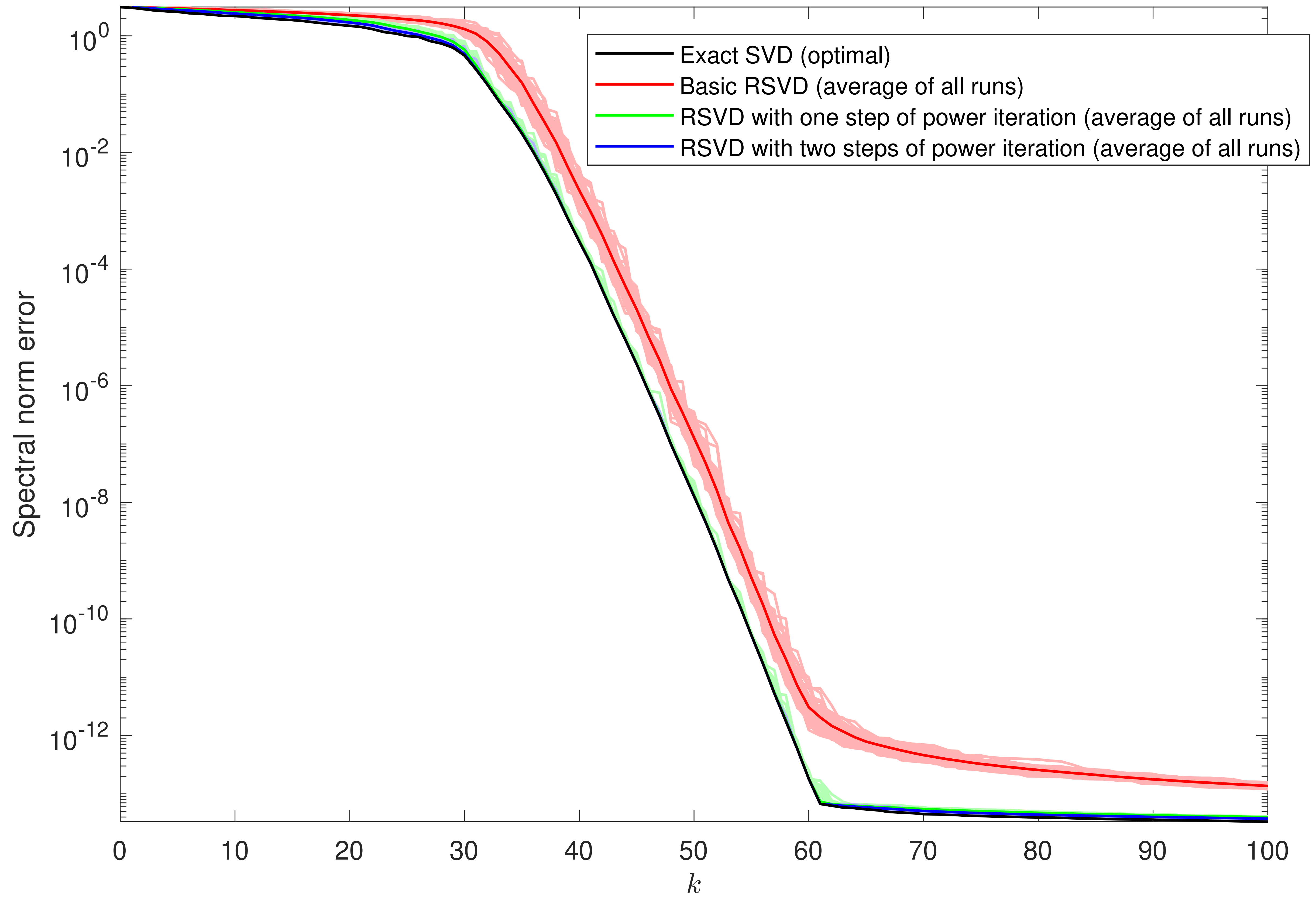
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

$$\mathbf{Y} = \left(\mathbf{A}\mathbf{A}^*\right)^q\mathbf{A}\mathbf{G},$$

and where $\mathbf{G}$ is a Gaussian random matrix.

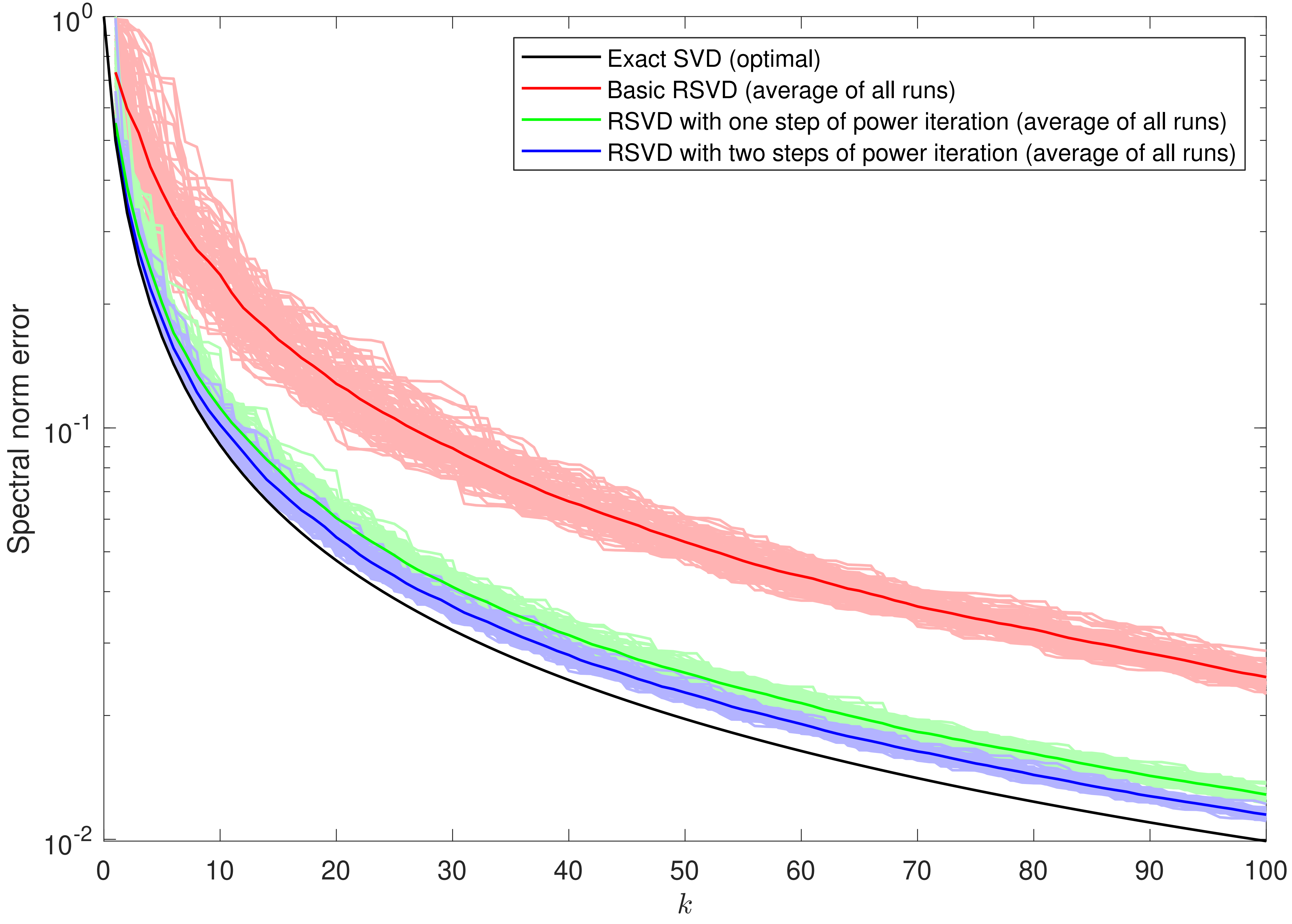
The matrix $\mathbf{A}$ now has singular values that decay slowly.

Randomized low rank approximation: The same plot, but showing 100 instantiations.



The darker lines show the mean errors across the 100 experiments.

Randomized low rank approximation: The same plot, but showing 100 instantiations.



The darker lines show the mean errors across the 100 experiments.

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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.	
(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Oversampling: By drawing a small number p of extra samples, we can prove that the error is close to theoretically minimal. Think $p = 5$ or $p = 10$.

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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Oversampling: By drawing a small number p of extra samples, we can prove that the error is close to theoretically minimal. Think $p = 5$ or $p = 10$. For instance, we have:

Theorem: [Halko, Martinsson, Tropp, 2009 & 2011] Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$. Let k denote a target rank and let p denote an over-sampling parameter. Let $\mathbf{G}$ denote an $n \times (k+p)$ Gaussian matrix. Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{A}\mathbf{G})$. If $p \geq 2$, then

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

and

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

There are also bounds on the error when power iteration is used, the likelihood of large deviations from the expected value, and so on.

Focus of current work is construction of *à posteriori* error bounds, and estimates on the accuracy of computed singular *vectors*. (With Yijun Dong and Yuji Nakatsukasa.)

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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.	
(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

- High practical speed — interacts with $\mathbf{A}$ only through matrix-matrix multiplication.

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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

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

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

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

(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 $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

- High practical speed — interacts with **A** only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).

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 $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AG}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

- High practical speed — interacts with $\mathbf{A}$ only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn\log k)$.

  The key is to use a *Fast Johnson-Lindenstrauss transform*.

  Practical acceleration is achieved at ordinary matrix sizes.

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

Output: Ra

(1) Draw a ... $\mathbf{Q}^*\,\mathbf{A}$.

(2) Form tl ... $\hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(3) Compu

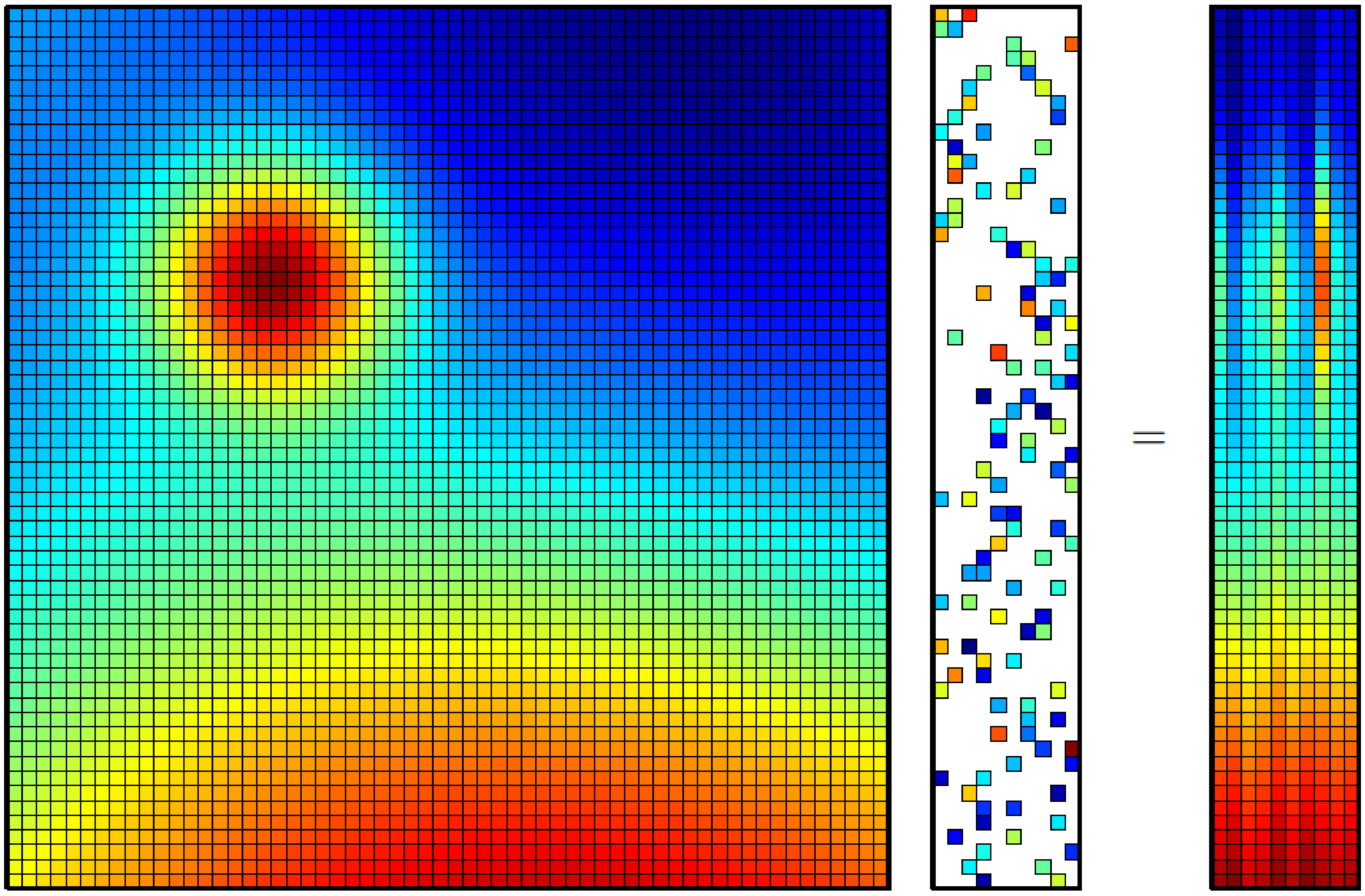
Key results

- High pra ... on.
- Order o
- Highly e
- Conside ... of a

  dense n ... k).

  The key

  Practica



The matrix **Ω** *is a sparse random matrix. Two nonzero entries are placed randomly in each row. In consequence, each column of* **A** *contributes to precisely two columns of the sample matrix* $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$*. This structured random map has* $O(mn)$ *complexity, is easy to work with practically, and often provides good accuracy.*

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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

- High practical speed — interacts with $\mathbf{A}$ only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn\log k)$.
- Single pass algorithms have been developed for *streaming environments.*

  The idea is that you are allowed to observe each matrix element only once.

  You cannot store the matrix.

  Not possible with deterministic methods!

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 $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AG}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

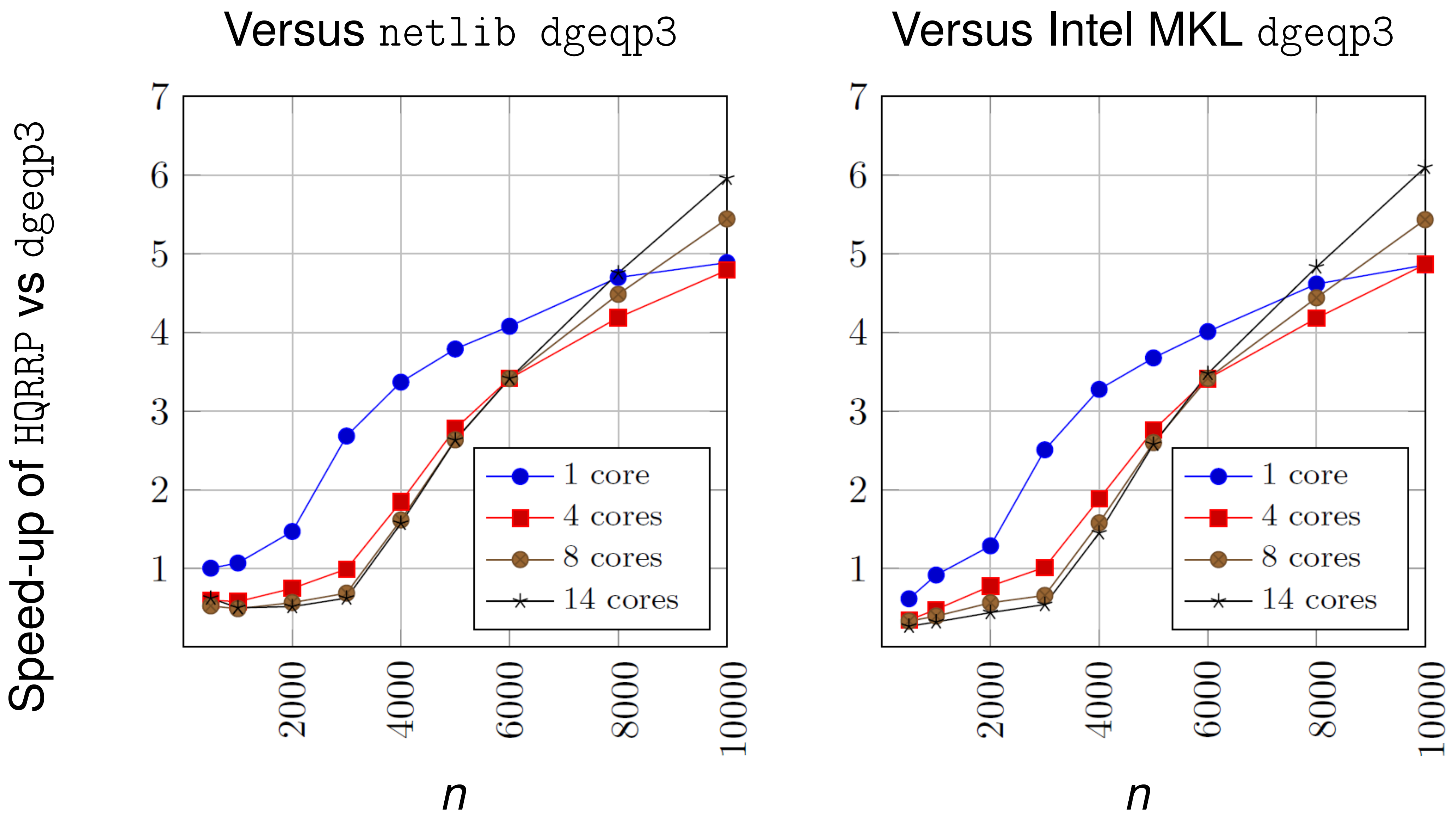
- High practical speed — interacts with $\mathbf{A}$ only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn\log k)$.
- Single pass algorithms have been developed for *streaming environments.*

  The idea is that you are allowed to observe each matrix element only once.

  You cannot store the matrix.

  Not possible with deterministic methods!
- The randomization idea can be used to overcome a classical problem in numerical linear algebra: How to efficiently implement column-pivoted QR factorizations. (How to cast the computation as BLAS3 operations instead of BLAS2.)

A very fast *randomized* implementation of column pivoted QR



Speedup attained by a randomized algorithm for computing a full column pivoted QR factorization of an $n \times n$ matrix. The speed-up is measured versus LAPACK's faster routine `dgeqp3` *as implemented in Netlib (left) and Intel's MKL (right). Our implementation was done in C, and was executed on an Intel Xeon E5-2695. Joint work with G. Quintana-Ortí, N. Heavner, and R. van de Geijn (SISC 2017). Closely related work by Duersch and Gu, SISC 2017 / SIREV 2020.*

References on randomized SVD:

- N. Halko, P. G. Martinsson, J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*, SIAM Review, 2011.
- P.G. Martinsson & J.A. Tropp, *Randomized Numerical Linear Algebra: Foundations & Algorithms*. 2020 Acta Numerica. arxiv #2002.01387.
- P.G. Martinsson, V. Rokhlin and M. Tygert (2006a), A randomized algorithm for the approximation of matrices, Technical Report Yale CS research report YALEU/DCS/RR-1361, Yale.
- Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, *Randomized algorithms for the low-rank approximation of matrices*. PNAS 2007 104: 20167-20172.
- F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis, **25**(3), 2008.
- V. Rokhlin, A. Szlam, and M. Tygert, *A Randomized Algorithm for Principal Component Analysis*, SIAM J. Matrix Anal. Appl., 31(3), 1100–1124, 2009.
- P.G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the approximation of matrices*. Applied and Computational Harmonic Analysis, 30(1), pp. 47–68, 2011.

Relevant prior work in:

- C. H. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala (2000), *Latent semantic indexing: a probabilistic analysis*, Vol. 61, pp. 217–235.
- A. Frieze, R. Kannan and S. Vempala (2004), *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. ACM 51(6), 1025–1041.

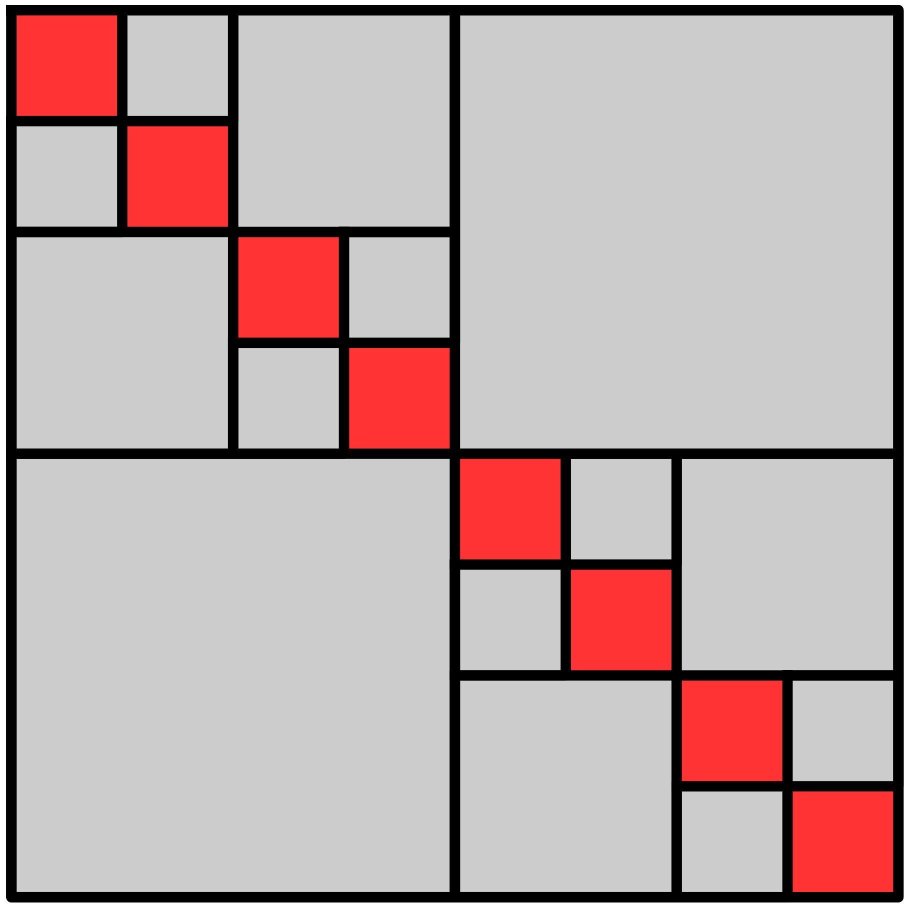
Outline of talk:

- Introduction: Problem formulation & solution operators. *[Done!]*
- Curse of dimensionality. *[Done!]*
- Interaction ranks — why are they small? How small are they? *[Done!]*
- (Versions of fast direct solvers — “strong” versus “weak” etc.) *[Skipped.]*
- High order discretizations and fast direct solvers. *[Done!]*
- Randomized low rank approximation. *[Done!]*
- Randomized compression of rank structured matrices.

Rank structured matrices

We use the term *rank structured* to refer to matrices that are not themselves of globally low rank, but can be tessellated into sub-blocks in such as way that each block is either small or of low numerical rank.

We focus on "hierarchical" tessellations (as the one shown on the right). Some techniques apply to "flat" formats as well.


All gray blocks have low rank.

Hierarchically rank structured matrices often admit linear or close to linear complexity algorithms for the matrix-vector multiply, matrix-matrix multiply, LU factorization, etc.

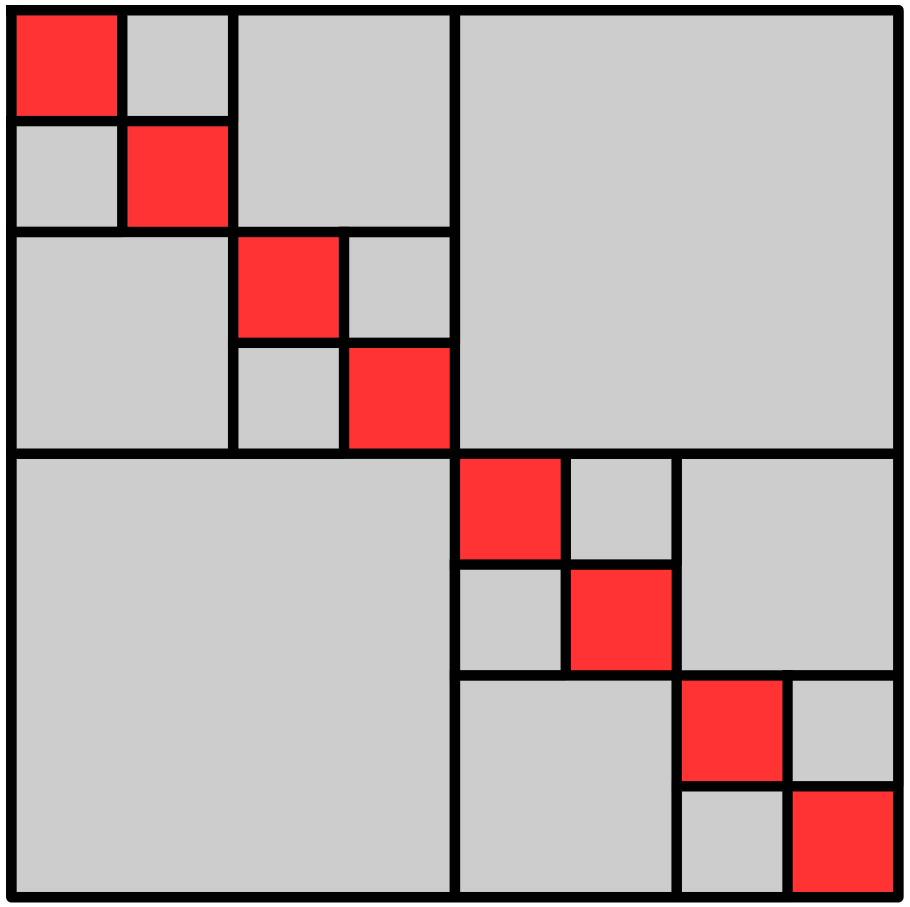
Ubiquitous applications in scientific computing: *Solution operators for elliptic PDEs, DtN operators, scattering matrices, Schur complements in sparse direct solvers, etc.*

More recently, have been shown to arise in data science as well — kernel matrices, covariance matrices, Hessians, etc.

Rank structured matrices

We use the term *rank structured* to refer to matrices that are not themselves of globally low rank, but can be tessellated into sub-blocks in such as way that each block is either small or of low numerical rank.

We focus on “hierarchical” tessellations (as the one shown on the right). Some techniques apply to “flat” formats as well.



All gray blocks have low rank.

Hierarchically rank structured matrices often admit linear or close to linear complexity algorithms for the matrix-vector multiply, matrix-matrix multiply, LU factorization, etc.

Ubiquitous applications in scientific computing: *Solution operators for elliptic PDEs, DtN operators, scattering matrices, Schur complements in sparse direct solvers, etc.*

References: Fast Multipole Method (Greengard, Rokhlin); Panel Clustering (Hackbusch); $\mathcal{H}$- and $\mathcal{H}^2$-matrices (Hackbusch et al); Hierarchically Block Separable matrices; Hierarchically Semi Separable matrices (Xia et al); HODLR matrices (Darve et al); BLR matrices (Buttari, Amestoy, Mary, ...); ...

In real life, tessellation patterns of rank structured matrices tend to be more complex . . .

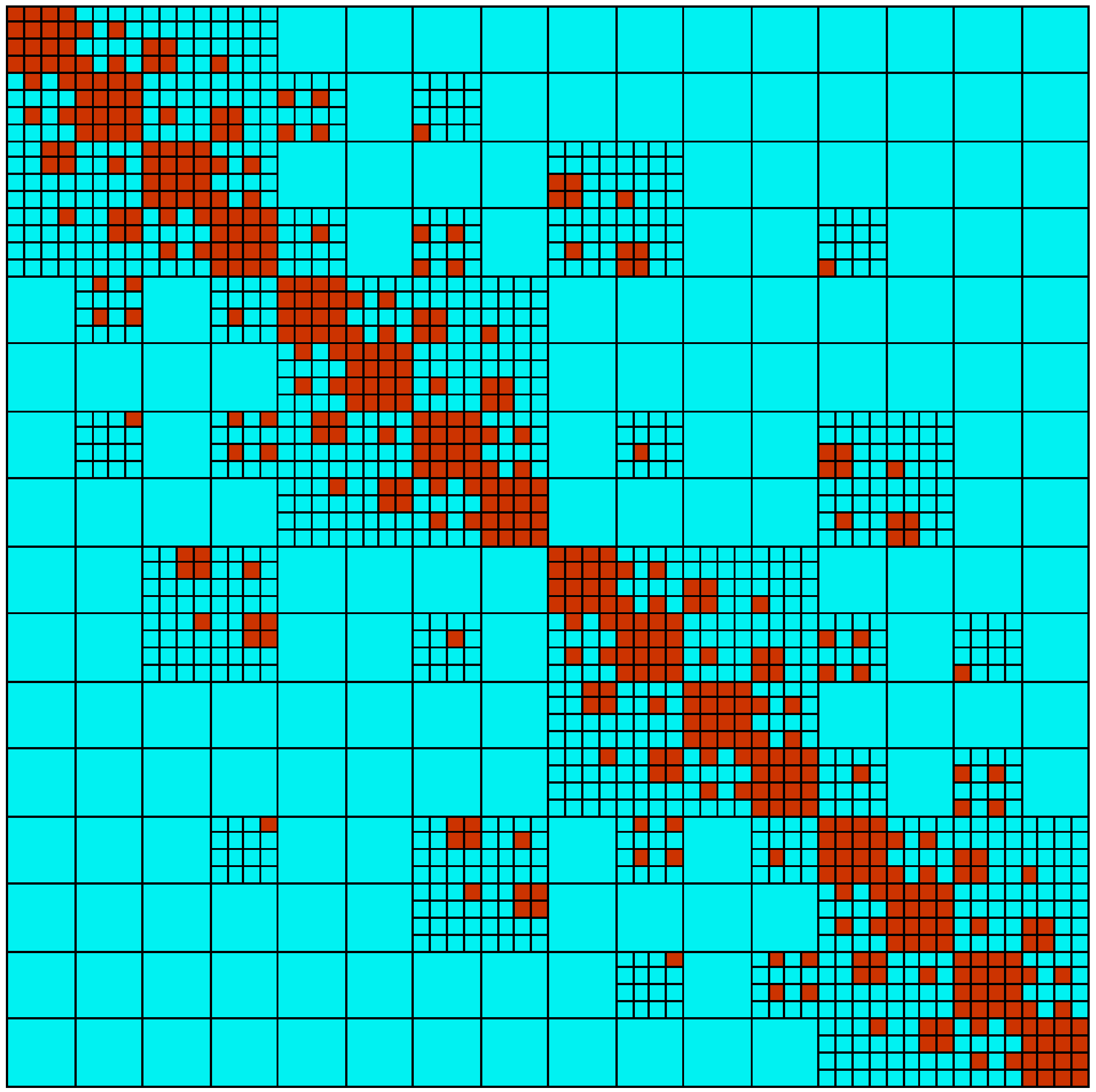


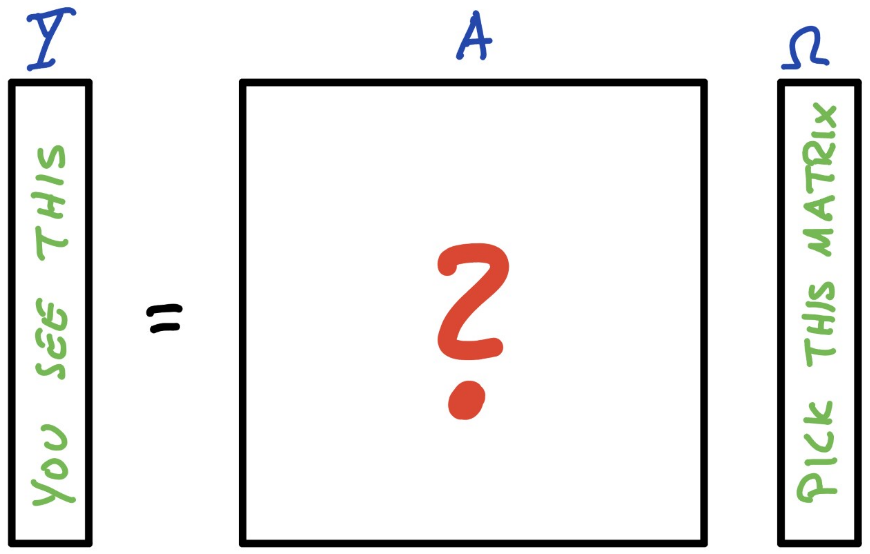
Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

Randomized approximation of rank structured matrices

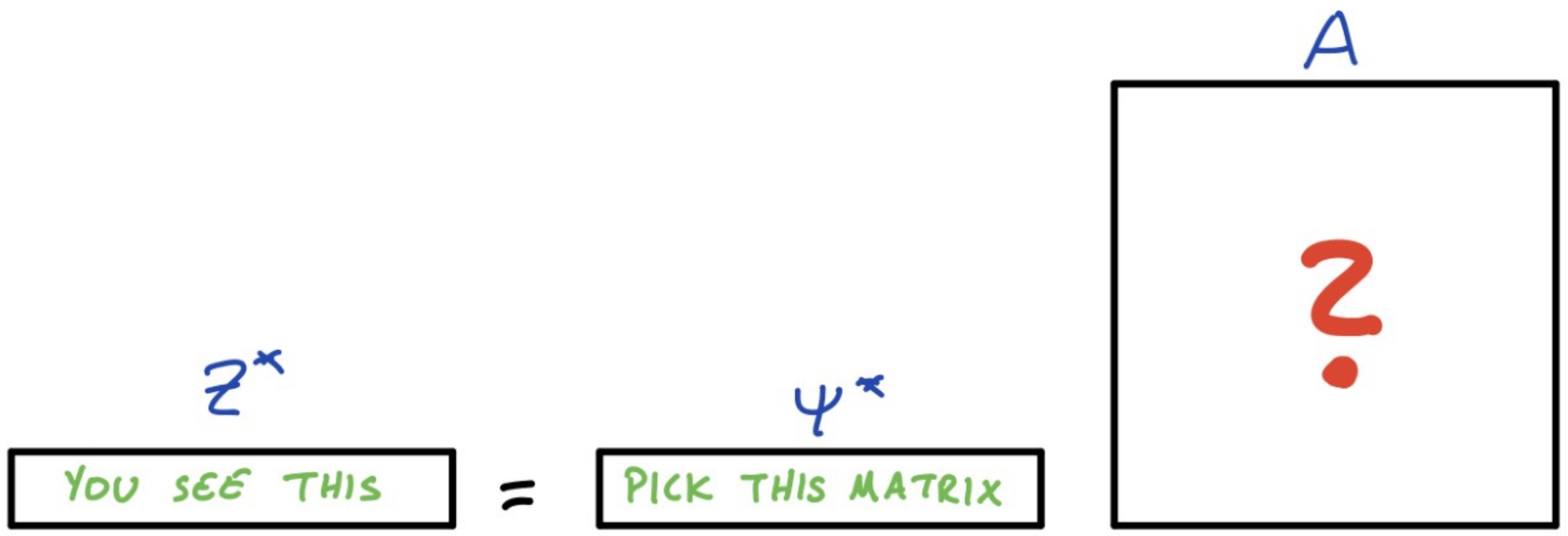
Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

Objective: Construct thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ such that $\mathbf{A}$ can be completely reconstructed in $O(N)$ work from the set $\{\mathbf{Y}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{\Psi}\}$ where $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

Sample the column space of the matrix:



If $\mathbf{A} \neq \mathbf{A}^*$, then sample the row space too:



Randomized approximation of rank structured matrices

Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

Objective: Construct thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ such that $\mathbf{A}$ can be completely reconstructed in $O(N)$ work from the set $\{\mathbf{Y}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{\Psi}\}$ where $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

The low rank case: In the particularly simple case where $\mathbf{A}$ has *global* rank k, we revert to the case we considered previously in the talk.

In the current framework, the randomized SVD takes the form:

- Set $s = k$ and draw a “test matrix” $\mathbf{\Omega} \in \mathbb{R}^{N \times s}$ from a Gaussian distribution.
- Form the “sample matrix” $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.
- Build $\mathbf{\Psi}$ to hold an ON basis for ran($\mathbf{Y}$), e.g., $[\mathbf{\Psi}, \sim] = \texttt{qr}(\mathbf{Y}, 0)$.
- Form $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$.

Then $\mathbf{A} = \mathbf{\Psi}\left(\mathbf{\Psi}^*\mathbf{A}\right) = \mathbf{\Psi}\mathbf{Z}^*$ with probability 1.

In the more typical case where $\mathbf{A}$ is only *approximately* of rank k, some *oversampling* is required to get a reliable scheme. (Say $s = k + 10$, or $s = 2k$, or some such.)

Randomized approximation of rank structured matrices

Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

Objective: Construct thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ such that $\mathbf{A}$ can be completely reconstructed in $O(N)$ work from the set $\{\mathbf{Y}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{\Psi}\}$ where $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

Why generalize from "global low rank" to "rank structured":

- Integral operators from classical physics. If you have a legacy method for the matrix-vector multiple (e.g. the Fast Multipole Method), then we could enable a range of operations – LU factorization, matrix inversion, etc.
- Multiplication of operators. Useful for forming Dirichlet-to-Neumann operators, for combining solvers of multi-physics problems, etc.
- Compression of Schur complements that arise in the LU or Cholesky factorization of sparse matrices. This lets us overcome key bottlenecks (e.g. LU factorization of a "finite element" matrix is accelerated from $O(N^2)$ to close to linear complexity.)

Randomized approximation of rank structured matrices

Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

Objective: Construct thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ such that $\mathbf{A}$ can be completely reconstructed in $O(N)$ work from the set $\{\mathbf{Y}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{\Psi}\}$ where $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

A variety of algorithms exist:

"Not quite black box": 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 in practice — requires only $O(k)$ matvecs. *P.G. Martinsson, SIMAX,* ***32****(4), 2011. Later improvements by Jianlin Xia, Sherry Li, and others.*

Peeling algorithms: True black box. But quite expensive in practice – require *many* matrix vector products. *L. Lin, J. Lu, L. Ying, JCP 2011. P.G. Martinsson, SISC,* ***38****(4), pp. A1959-A1986, 2016.*

James Levitt dissertion: First (?) true black box and true linear complexity methods. Introduces several new ideas, including "block nullification" and "block extraction". *J. Levitt & P.G. Martinsson, arxiv arXiv:2205.02990, 2022. J. Levitt dissertation at* https://users.oden.utexas.edu/∼pgm/main_publications.html

Randomized strong recursive skeletonization: Employs the block nullification and block extraction ideas to the task of compressing all intermediate matrices in the strong recursive skeletonization procedure. *A. Yesypenko, P.G. Martinsson, arXiv:2311.01451.*

Core idea or RSRS

Suppose you have extracted samples

$$\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega} \qquad \text{and} \qquad \mathbf{Z} = \mathbf{A}^*\boldsymbol{\Psi}.$$

We use the set $\{\mathbf{Y}, \boldsymbol{\Omega}, \mathbf{Z}, \boldsymbol{\Psi}\}$ to extract the information to compress the first block using "block nullification" and "block extraction".

Then do one step of strong recursive skeletonization to obtain a partial factorization

$$\mathbf{A} = \mathbf{L}\tilde{\mathbf{A}}\mathbf{R},$$

where $\mathbf{L}$ and $\mathbf{R}$ each consists of two block elimination steps, and $\tilde{\mathbf{A}}$ is a matrix where some blocks have been zeroed out, and some have been modified.

We next seek a sample of $\tilde{\mathbf{A}}$. To do this, we form

$$\tilde{\mathbf{Y}} := \mathbf{L}^{-1}\mathbf{Y} = \mathbf{L}^{-1}\mathbf{A}\boldsymbol{\Omega} = \mathbf{L}^{-1}\big(\mathbf{L}\tilde{\mathbf{A}}\mathbf{R}\big)\boldsymbol{\Omega} = \tilde{\mathbf{A}}\tilde{\boldsymbol{\Omega}},$$

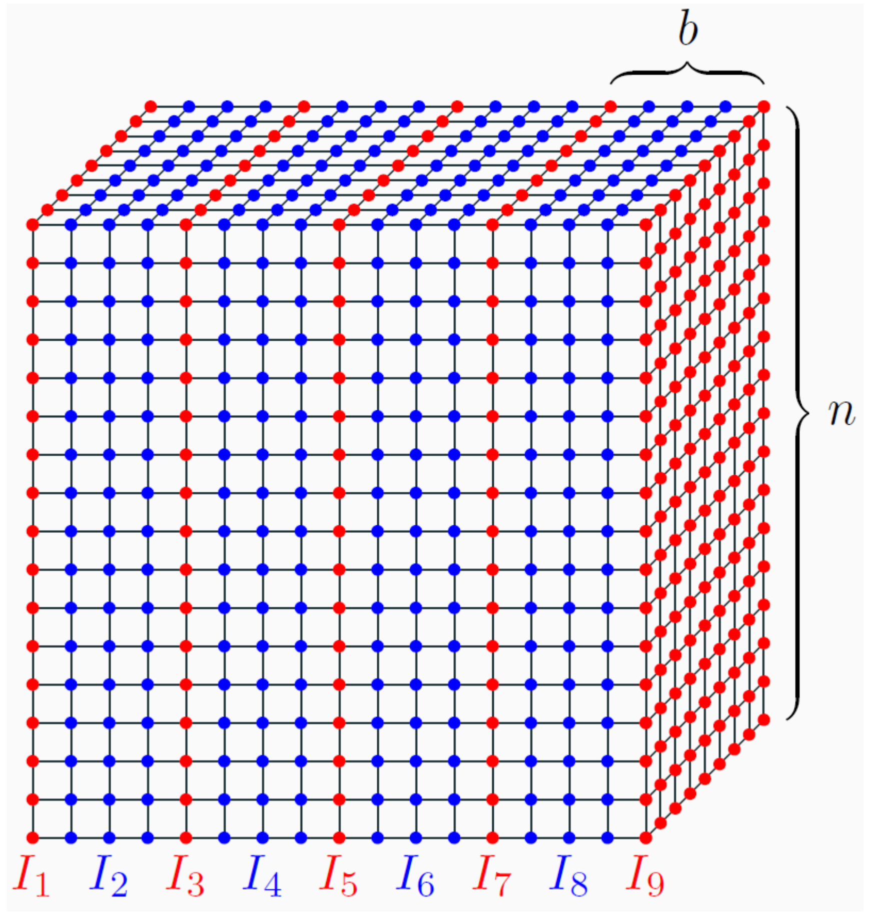
where we defined

$$\tilde{\boldsymbol{\Omega}} := \mathbf{R}\boldsymbol{\Omega}.$$

Analogously, form $\{\tilde{\mathbf{Z}},\, \tilde{\boldsymbol{\Psi}}\}$.

Proceed to the next box using the set $\{\tilde{\mathbf{Y}}, \tilde{\boldsymbol{\Omega}}, \tilde{\mathbf{Z}}, \tilde{\boldsymbol{\Psi}}\}$ in place of $\{\mathbf{Y}, \boldsymbol{\Omega}, \mathbf{Z}, \boldsymbol{\Psi}\}$.

Application of RSRS: A slab based LU factorization algorithm

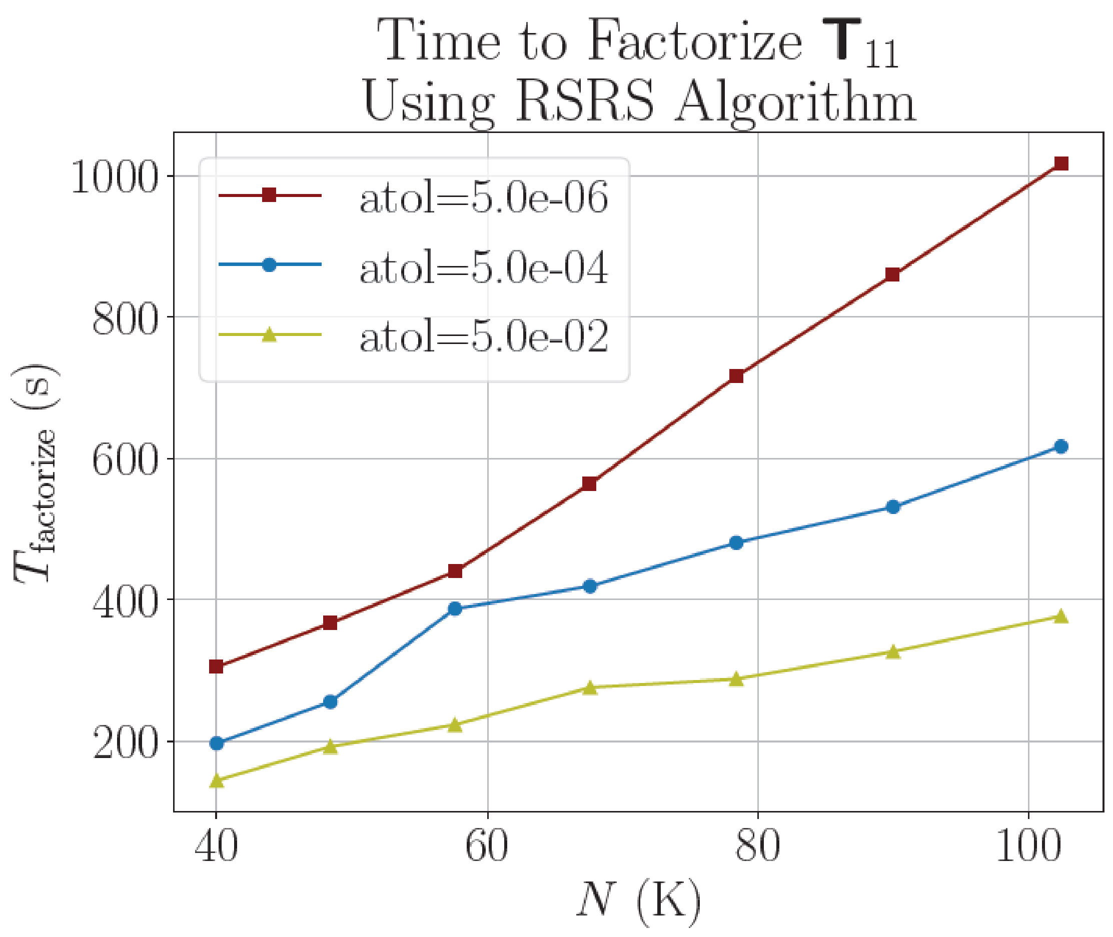


Our solver slabLU has the following characteristics:

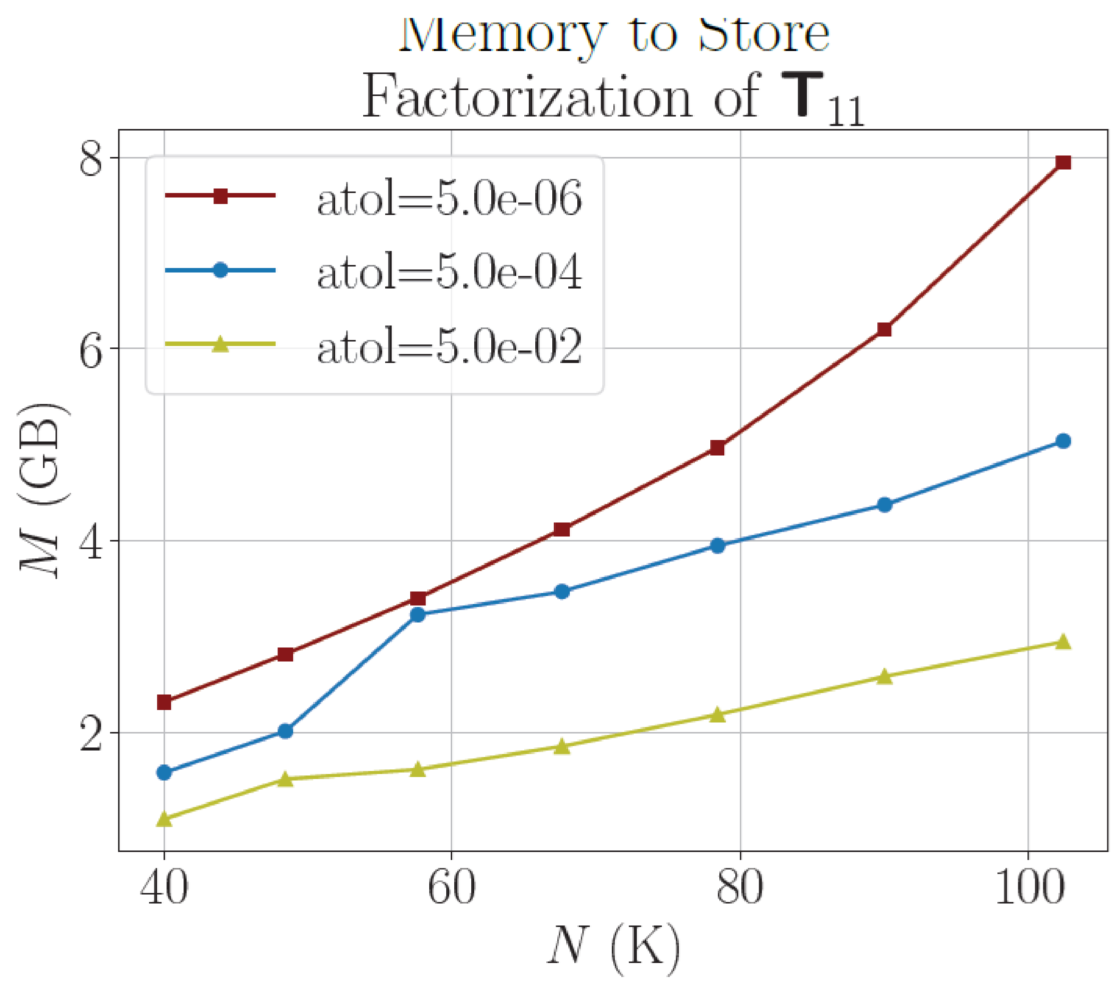
- The domain is partitioned into thin “slabs”.
- The thinness of each slab is exploited by a local sparse direct solver to eliminate the interior nodes in each slab (blue in the figure).
- RSRS is used to construct and factorize the Schur complements that “live” on the slab interfaces (red in the figure).
- RSRS is then again used to solve the resulting block tridiagonal system.

The 2D version can handle a domain of size $900\lambda \times 900\lambda$ (using 80M dofs) on a desktop. 10min for factorization. 40sec for solve. Three digits of relative accuracy *in the solution.*

RSRS within 3D slabLU: Numerical results

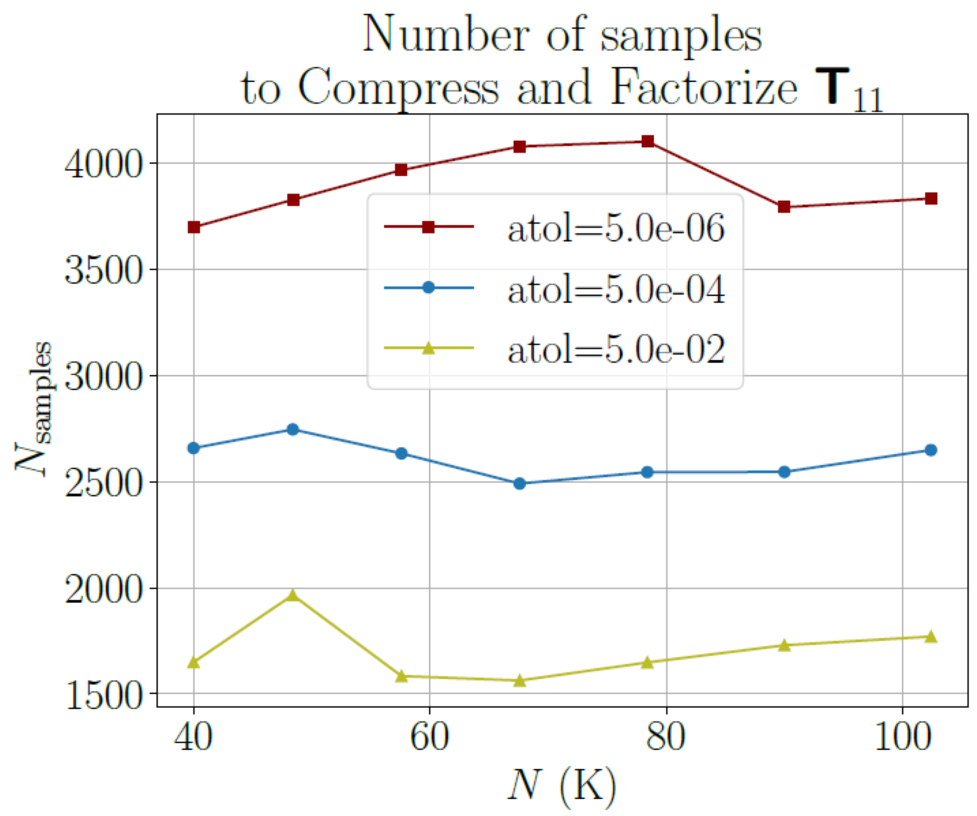


(A) Time to compute $\mathbf{T}_{11}^{-1}$ scales linearly.

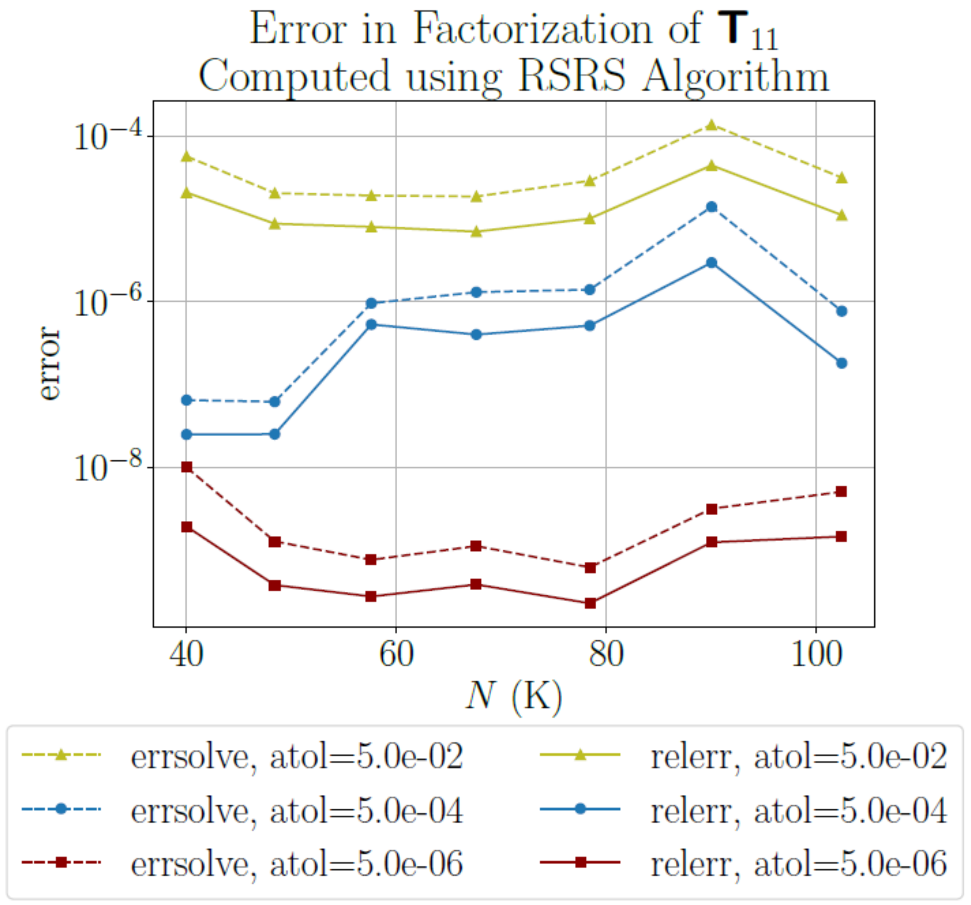


(B) Memory to store $\mathbf{T}_{11}^{-1}$ scales linearly.

RSRS within 3D slabLU: Numerical results



(c) Number of samples needed is independent of N.

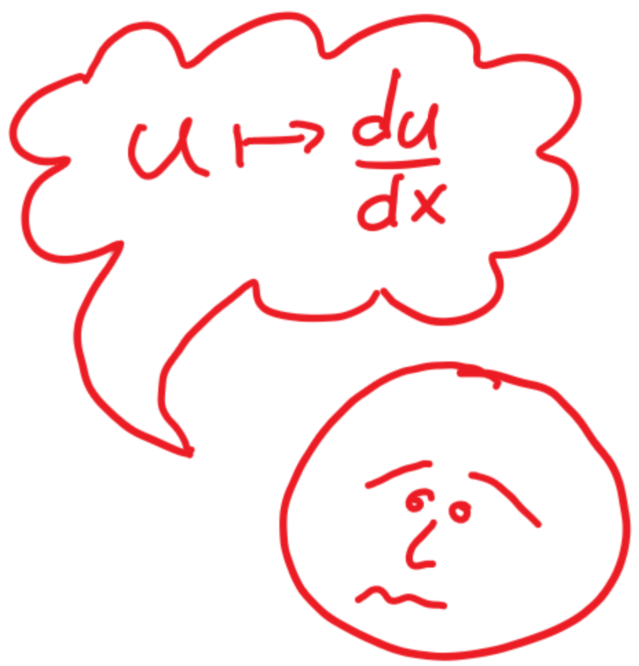


(d) The computed factorization is accurate and does not deteriorate with increasing N.

Key Ideas:

The solution operator of a linear elliptic PDE is "friendly."

- Smoothing.
- Stable.



$u \mapsto \int u$

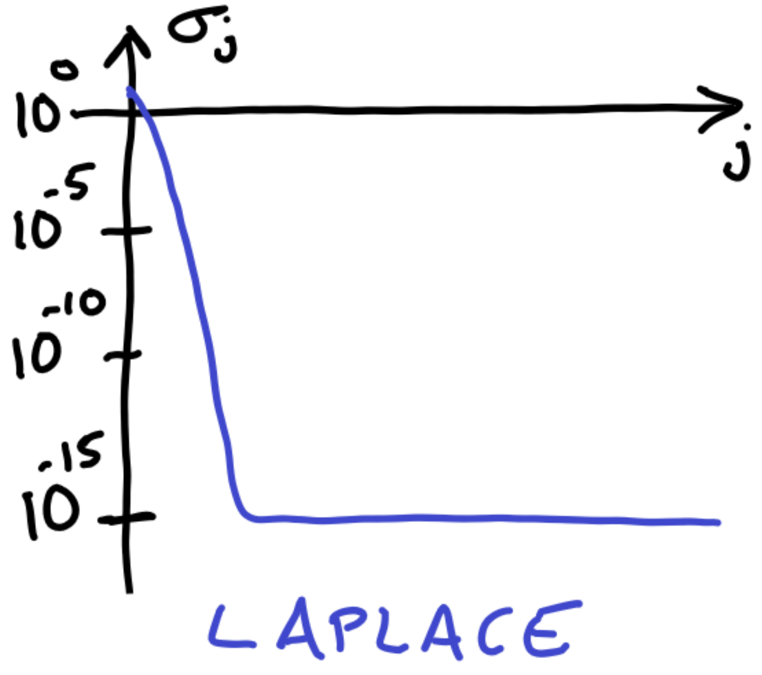
Key Ideas:

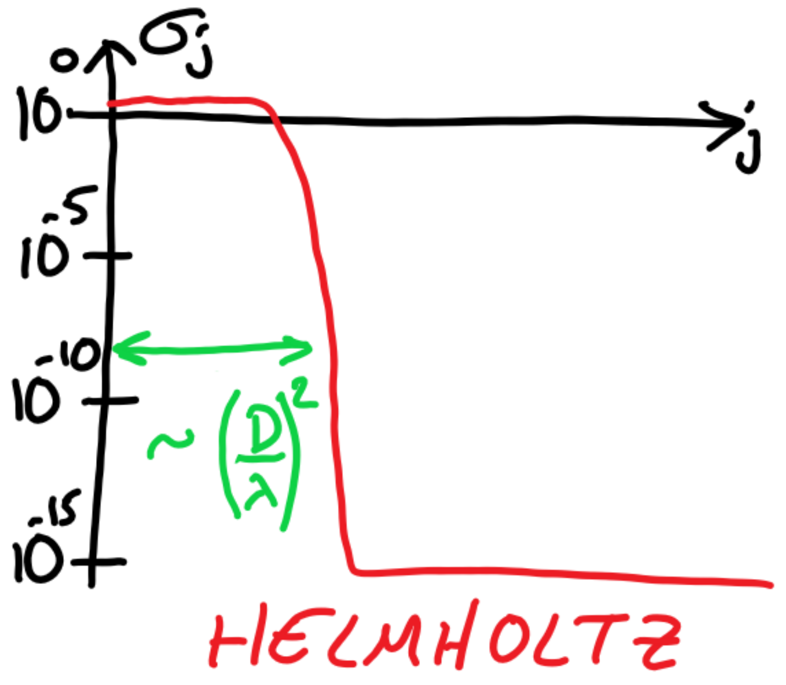
The solution operator of a linear elliptic PDE is "friendly."

- Smoothing.
- Stable.

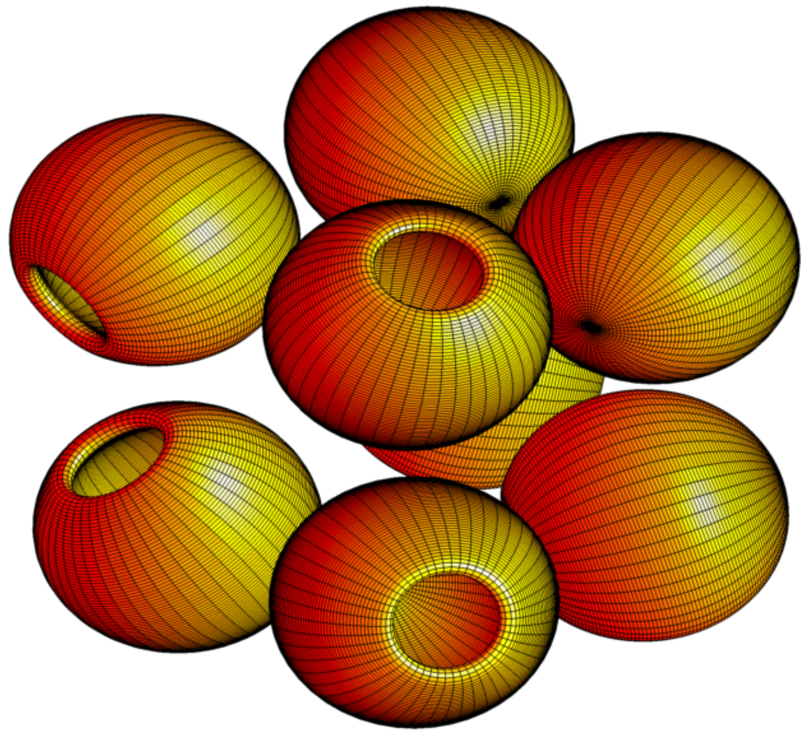
Long range interactions are low rank.

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





Key Ideas:

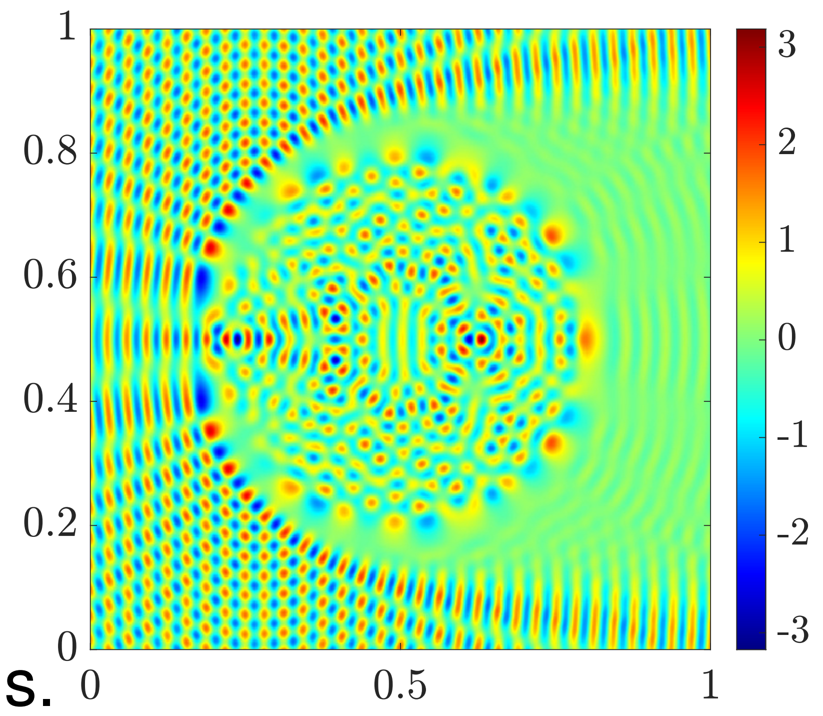


The solution operator of a linear elliptic PDE is "friendly."

- 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.



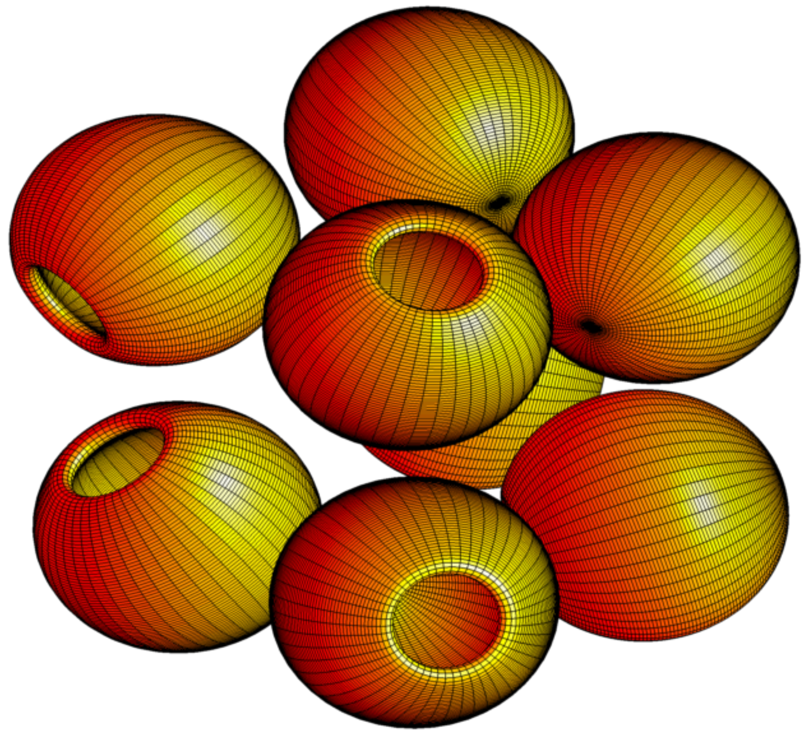
High-order discretizations + FDS.

- Maximize the work done by each DOF to save memory.
- Perfect for ill-conditioned problems with oscillatory solutions.
- Requires care in choosing discretization scheme.

Key Ideas:

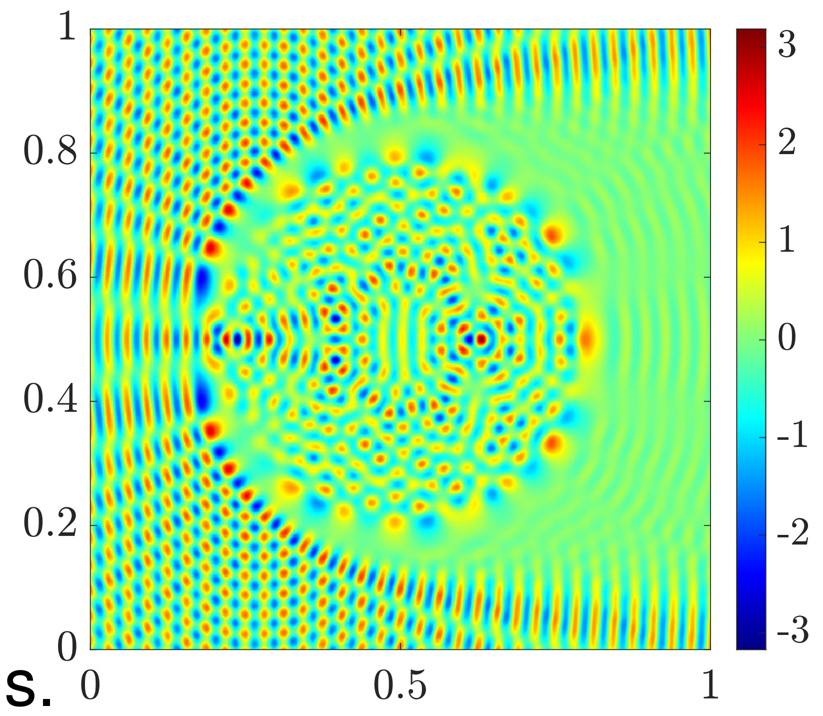
The solution operator of a linear elliptic PDE is "friendly."

- 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.



High-order discretizations + FDS.

- Maximize the work done by each DOF to save memory.
- Perfect for ill-conditioned problems with oscillatory solutions.
- Requires care in choosing discretization scheme.

New randomized methods for matrix algebra $\rightarrow$ acceleration & simplification.

Another talk: Extensive applications in data science and computational statistics.

Where we are now:

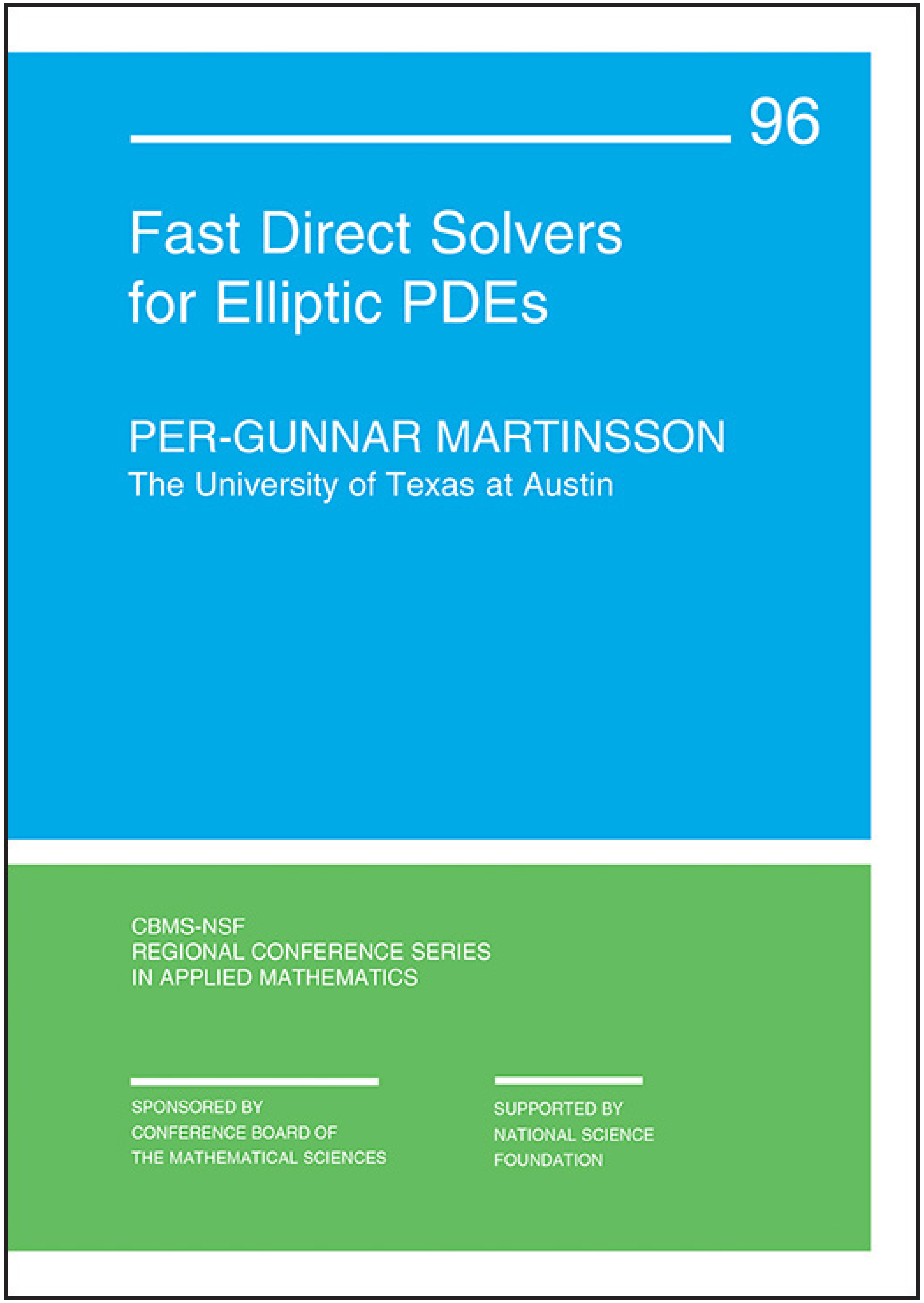
- We have developed direct solvers with $O(N)$ complexity for elliptic PDEs with non-oscillatory (or "mildly oscillatory") solutions for most standard environments:
  - Sparse matrices from FEM/FD/composite spectral/... 3D work is in progress.
  - Boundary integral equations in 2D and 3D. (Acceleration of 3D solvers is 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 $\rightarrow$ 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!)*
- Parabolic and hyperbolic problems. Parallel-in-time methods?
- Operator algebra. Multiphysics problems. FEM-BEM coupling.

More details in a 2019 monograph:



96

Fast Direct Solvers for Elliptic PDEs

PER-GUNNAR MARTINSSON
The University of Texas at Austin

CBMS-NSF
REGIONAL CONFERENCE SERIES
IN APPLIED MATHEMATICS

SPONSORED BY
CONFERENCE BOARD OF
THE MATHEMATICAL SCIENCES

SUPPORTED BY
NATIONAL SCIENCE
FOUNDATION