SIAM CSE 2025: Mini-tutorial 4

Fast Direct Solvers for Elliptic BIEs

Adrianna Gillman, University of Colorado at Boulder

Gunnar Martinsson, University of Texas at Austin

Research support by:

Problem addressed: The tutorial describes numerical methods for solving 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 tutorial describes numerical methods for solving 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:

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

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

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

Problem addressed: The tutorial describes numerical methods for solving 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 tutorial describes numerical methods for solving 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.*

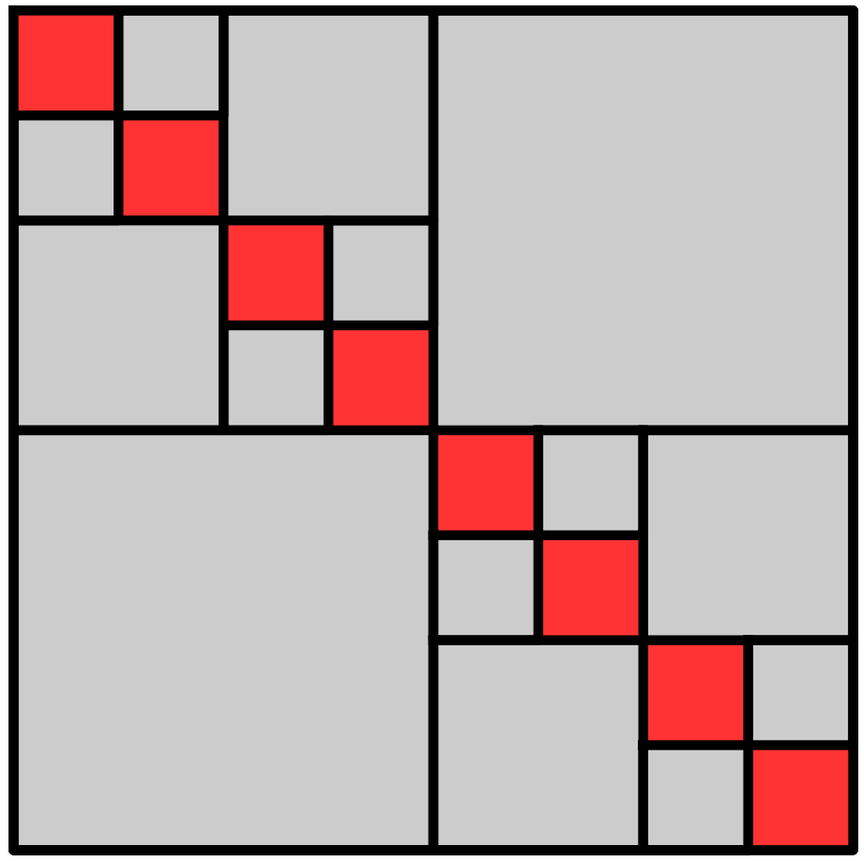
Dense matrices upon discretization. $O(N^2)$ cost? $O(N^3)$ cost?

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 ...

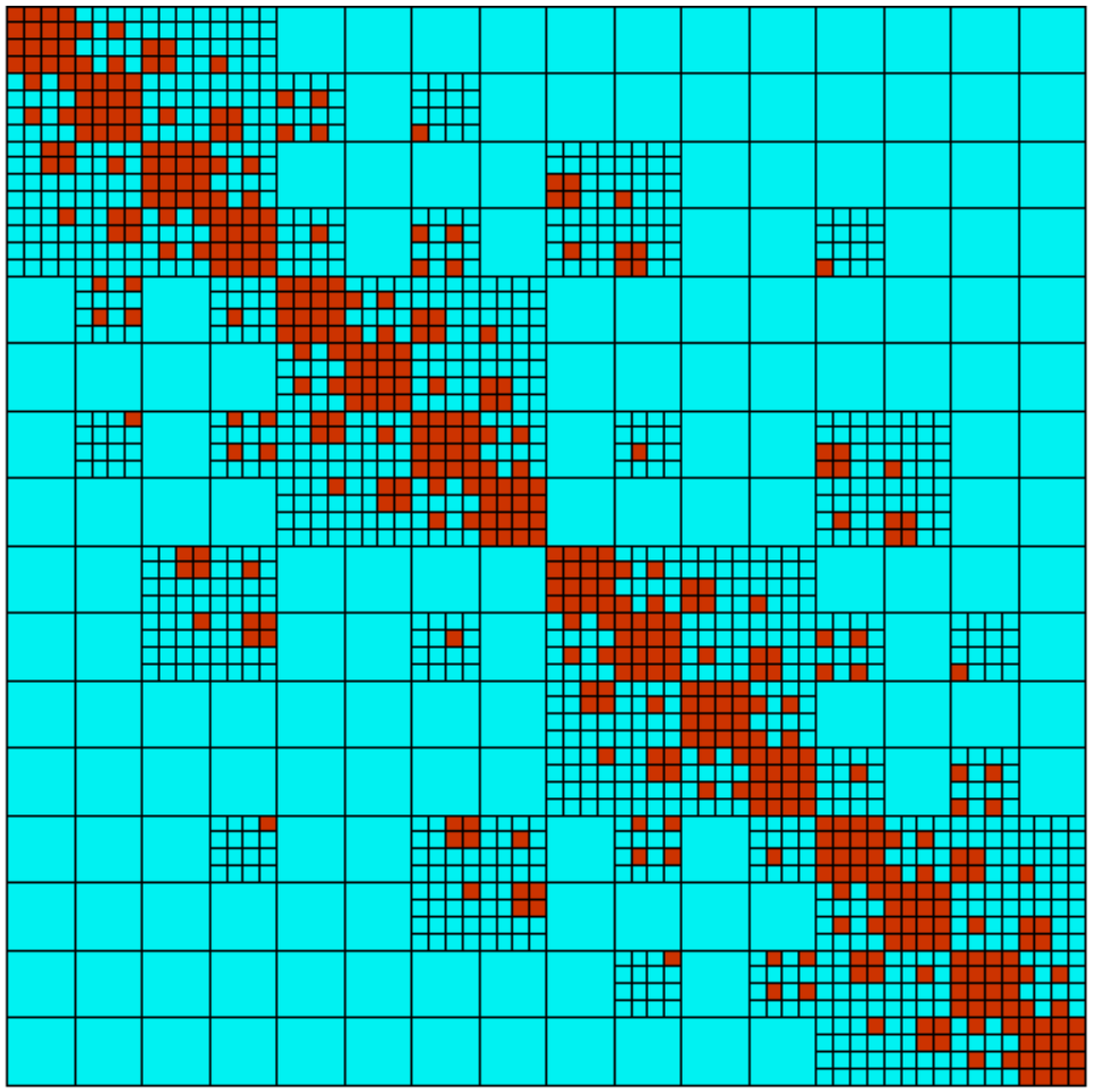


Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

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

Partial Differential Equation (PDE) vs. Integral Equation (IE) formulations

In the first tutorial, the focus is on the classical PDE framework:
Discretize using, say, finite elements; obtain a linear system $\mathbf{Au} = \mathbf{b}$ for some *sparse* matrix $\mathbf{A}$, then build an invertible factorization of $\mathbf{A}$.

In the second tutorial, the focus shifts in part to an IE framework:
Reformulate the PDE as an integral equation using analysis, then discretize the IE using, say, Nyström; obtain a linear system $\mathbf{Au} = \mathbf{b}$ for some *dense* matrix $\mathbf{A}$, then build an invertible factorization of $\mathbf{A}$.

In the context of direct solvers, dense matrices are unavoidable, and must be handled.
This makes IE formulations compelling, as they lead to better conditioned linear systems.

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)

The tutorial 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.

Advantages of direct solvers

1. They enable the solution of problems intractable to iterative methods:
   - Scattering problems near resonant frequencies.
   - Ill-conditioning due to geometry (elongated domains, percolation, etc).
   - Multi-physics simulations.
   - Mixed discretizations, say combining finite elements and boundary element methods.

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

2. They accelerate computations in applications that require a very large number of solves for a fixed operator:
   - Scattering problems.
   - Time stepping of parabolic PDEs.
   - Optimal design. (Local updates to the system matrix are cheap.)

   A couple of orders of magnitude speed-up is sometimes possible.

3. They open the door to *operator algebra.* *Next few slides.*
   - Composition of operators.
   - Exponentiation (and other functions).
   - Full and partial spectral decompositions.

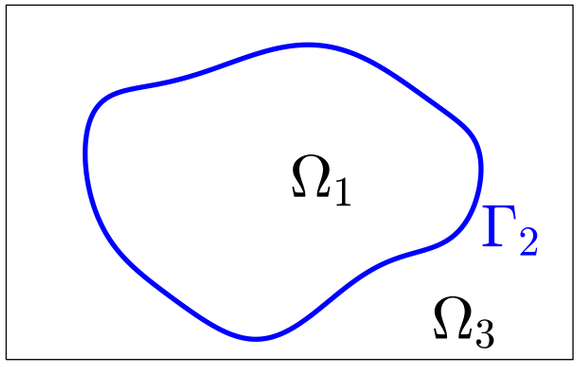
Operator algebra: Composition of operators

The algorithms for computing an approximation to an inverse A^{-1} of a differential or integral operator A also allow us to compute *compositions* of operators such as

$$F = AB, \qquad F = A^{-1}B \qquad F = D - BA^{-1}C, \qquad \ldots$$

Example: Consider a simple problem involving two interacting domains, e.g., a wave propagation problem involving both a fluid and a solid medium:

The geometry:



The linear system:

$$\begin{array}{|ccc||c|}
\hline
A_{11} & A_{12} & 0 & \\
\hline
A_{21} & A_{22} & A_{23} & = \\
\hline
0 & A_{32} & A_{33} & \\
\hline
\end{array}
\quad
\begin{array}{|c|}
\hline
x \\
\hline
\\
\hline
\end{array}
\quad
\begin{array}{|c|}
\hline
b \\
\hline
\\
\hline
\end{array}$$

Think of A_{11} and A_{33} as representing equilibrium operators for the subdomains Ω_1 and Ω_3. We seek to solve the full system that incorporates the interface conditions along Γ_2. In principle, we can reduce the problem to one that lives on the joint interface alone

$$(3) \qquad (A_{22} - A_{21}A_{11}^{-1}A_{12} - A_{23}A_{33}^{-1}A_{32})x_2 = b_2 - A_{21}A_{11}^{-1}b_1 - A_{23}A_{33}^{-1}b_3.$$

Existing methods rely on the ability to apply A_{11}^{-1} and A_{33}^{-1} efficiently via, e.g., multigrid, and then solve (3) iteratively. Using direct solvers, we can *explicitly* form and invert the dense operator $A_{22} - A_{21}A_{11}^{-1}A_{12} - A_{23}A_{33}^{-1}A_{32}$.

Operator algebra: Composition of operators

The algorithms for computing an approximation to an inverse A^{-1} of a differential or integral operator A also allow us to compute *compositions* of operators such as

$$F = AB, \qquad F = A^{-1}B \qquad F = D - BA^{-1}C, \qquad \ldots$$

The ability to explicitly form Schur complements and other compositions of operators arise in a range of different applications:

- Multi-physics simulations.
- Solving PDEs with constraints.
- Multi-discretizations. For instance, coupling FEM and BEM discretizations.
- Domain decomposition.

In these applications, the operators we seek to approximate are often discrete equivalences of Dirichlet-to-Neumann operators, Impedance-to-Impedance operators, or other Poincaré-Steklov operators.

Operator algebra: Exponentiation (and other functions)

It is often valuable to be able to explicitly compute a function $f(A)$ of a given elliptic operator A.

- *Time evolution operators for elliptic PDEs:* Evaluating $f(A) = e^{-tA}$ allows for either taking long time-steps, or for very rapid time-stepping for short time steps.
- *Time evolution operators for hyperbolic PDEs:* Evaluating $f(A) = e^{it\sqrt{A}}$ allows for parallel-in-time time stepping.
- *Solving fractional pdes:* Evaluating $f(A) = A^{-\alpha}$ for $\alpha \in (0, 1)$ allows for direct solution of elliptic PDEs such as $(-\Delta)^{\alpha} u = f$. Similarly, $f(A) = e^{-A^{\alpha} t}$ is the time evolution operator for a fractional parabolic PDE.

Idea: A common technique is to build a rational approximation

$$f(t) \approx \sum_{p=1}^{P} \frac{a_p}{1 - b_p t}$$

that is valid to high accuracy on the "relevant" part of the spectrum of A. Then

$$f(A) \approx \sum_{p=1}^{P} a_p \left(I - b_p A\right)^{-1}$$

can be evaluated by solving a sequence of shifted elliptic PDEs.

Operator algebra: Spectral decompositions

In many applications, we seek to compute eigenvalues and eigenvectors of a given elliptic operator A:

- Compute resonance frequencies for vibrating systems.
- Computational chemistry and physics.
- Finding buckling modes of elasto-static structures.

When a small number of eigenmodes are sought, *shifted inverse iteration* is often a powerful tool. Given an approximate normalized eigenvector u_i, we proceed as follows:

$$\mu_i = (u_i, Au_i), \qquad u'_{i+1} = (A - \mu_i I)^{-1} u_i, \qquad u_{i+1} = \frac{1}{\|u'_{i+1}\|} u'_{i+1}.$$

Cauchy integrals are another useful tool. For instance, spectral projections can be computed via formulas such as

$$Pu = \left(\frac{1}{2\pi i} \int_\Gamma (z\, I - A)^{-1}\, dz \right) u \approx \sum_{p=1}^{P} w_p \frac{1}{2\pi i} (z_p\, I - A)^{-1} u.$$

In some case, one can even construct full unitary diagonalizations of structured matrices.

Introduction to rank structured matrices

How is it that you can invert a dense matrix of size $n \times n$ in less than $O(n^3)$ flops?

Inversion of structured matrices: Identity plus low rank

As a warm-up, let us consider a matrix of the form *identity + low rank.*

To be precise, suppose that a dense $n \times n$ matrix $\mathbf{A}$ is invertible, and that for some $k \ll n$, it admits the representation

$$\underset{n \times n}{\mathbf{A}} = \underset{n \times n}{\mathbf{I}} + \underset{n \times k}{\mathbf{U}} \underset{k \times n}{\mathbf{V}^*}.$$

Inversion of structured matrices: Identity plus low rank

As a warm-up, let us consider a matrix of the form *identity + low rank.*

To be precise, suppose that a dense $n \times n$ matrix $\mathbf{A}$ is invertible, and that for some $k \ll n$, it admits the representation

$$\underset{n \times n}{\mathbf{A}} = \underset{n \times n}{\mathbf{I}} + \underset{n \times k}{\mathbf{U}} \underset{k \times n}{\mathbf{V}^*}.$$

While $\mathbf{A}$ is dense, it has *rank structure* that allows us to:

- Store it using $O(nk)$ floats. *$O(n^2)$ in general case.*
- Execute the matrix vector multiplication in $O(nk)$ flops. *$O(n^2)$ in general case.*

Inversion of structured matrices: Identity plus low rank

As a warm-up, let us consider a matrix of the form *identity + low rank.*

To be precise, suppose that a dense $n \times n$ matrix $\mathbf{A}$ is invertible, and that for some $k \ll n$, it admits the representation

$$\underset{n\times n}{\mathbf{A}} = \underset{n\times n}{\mathbf{I}} + \underset{n\times k}{\mathbf{U}}\ \underset{k\times n}{\mathbf{V}^*}.$$

Then the inverse of $\mathbf{A}$ takes the form

$$\text{(W)} \qquad \underset{n\times n}{\mathbf{A}^{-1}} = \underset{n\times n}{\mathbf{I}} - \underset{n\times k}{\mathbf{U}}\ \underset{k\times k}{\left(\mathbf{I}+\mathbf{V}^*\mathbf{U}\right)^{-1}}\ \underset{k\times n}{\mathbf{V}^*}.$$

Using (W), we can form $\mathbf{A}^{-1}$ (implicitly) in $O(nk^2)$ work, instead of $O(n^3)$.

Inversion of structured matrices: Identity plus low rank

As a warm-up, let us consider a matrix of the form *identity + low rank.*

To be precise, suppose that a dense $n \times n$ matrix $\mathbf{A}$ is invertible, and that for some $k \ll n$, it admits the representation

$$\underset{n\times n}{\mathbf{A}} = \underset{n\times n}{\mathbf{I}} + \underset{n\times k}{\mathbf{U}}\ \underset{k\times n}{\mathbf{V}^*}.$$

Then the inverse of $\mathbf{A}$ takes the form

$$\text{(W)} \qquad \underset{n\times n}{\mathbf{A}^{-1}} = \underset{n\times n}{\mathbf{I}} - \underset{n\times k}{\mathbf{U}}\ \underset{k\times k}{\left(\mathbf{I}+\mathbf{V}^*\mathbf{U}\right)^{-1}}\ \underset{k\times n}{\mathbf{V}^*}.$$

Using (W), we can form $\mathbf{A}^{-1}$ (implicitly) in $O(nk^2)$ work, instead of $O(n^3)$.

There are many versions of the Woodbury formula:

Inversion of structured matrices: Identity plus low rank

As a warm-up, let us consider a matrix of the form *identity + low rank.*

To be precise, suppose that a dense $n \times n$ matrix $\mathbf{A}$ is invertible, and that for some $k \ll n$, it admits the representation

$$\underset{n\times n}{\mathbf{A}} = \underset{n\times n}{\mathbf{I}} + \underset{n\times k}{\mathbf{U}}\ \underset{k\times n}{\mathbf{V}^*}.$$

Then the inverse of $\mathbf{A}$ takes the form

$$\text{(W)} \qquad \underset{n\times n}{\mathbf{A}^{-1}} = \underset{n\times n}{\mathbf{I}} - \underset{n\times k}{\mathbf{U}}\ \underset{k\times k}{\left(\mathbf{I}+\mathbf{V}^*\mathbf{U}\right)^{-1}}\ \underset{k\times n}{\mathbf{V}^*}.$$

Using (W), we can form $\mathbf{A}^{-1}$ (implicitly) in $O(nk^2)$ work, instead of $O(n^3)$.

There are many versions of the Woodbury formula: For instance, consider a matrix of the form *"easy to invert" + low rank.* To be precise, suppose that $\mathbf{A}$ is invertible, that

$$\underset{n\times n}{\mathbf{A}} = \underset{n\times n}{\mathbf{D}} + \underset{n\times k}{\mathbf{U}}\ \underset{k\times n}{\mathbf{V}^*},$$

and that you can easily compute $\mathbf{D}^{-1}$. Then $\mathbf{A}^{-1}$ can be constructed via

$$\underset{n\times n}{\mathbf{A}^{-1}} = \underset{n\times n}{\mathbf{D}^{-1}} - \underset{n\times k}{\mathbf{D}^{-1}\mathbf{U}}\ \underset{k\times k}{\left(\mathbf{I}+\mathbf{V}^*\mathbf{D}^{-1}\mathbf{U}\right)^{-1}}\ \underset{k\times n}{\mathbf{V}^*\mathbf{D}^{-1}}.$$

Inversion of structured matrices: Tridiagonal

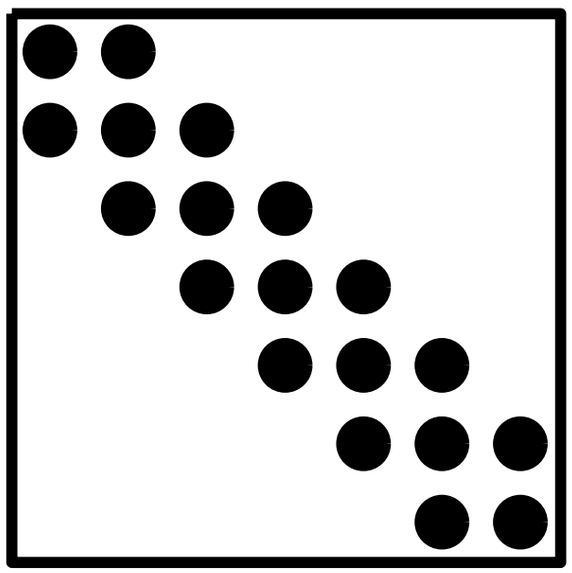
Consider a simple 2-point BVP on the interval $[0, 1]$:

$$\text{(BVP)} \qquad \begin{cases} -\dfrac{d^2u}{dx^2} + p(x)\dfrac{du(x)}{dx} + m(x)\,u(x) = g(x), & x \in (0,1), \\ u(0) = f_{\mathrm{L}}, \\ u(1) = f_{\mathrm{R}}. \end{cases}$$

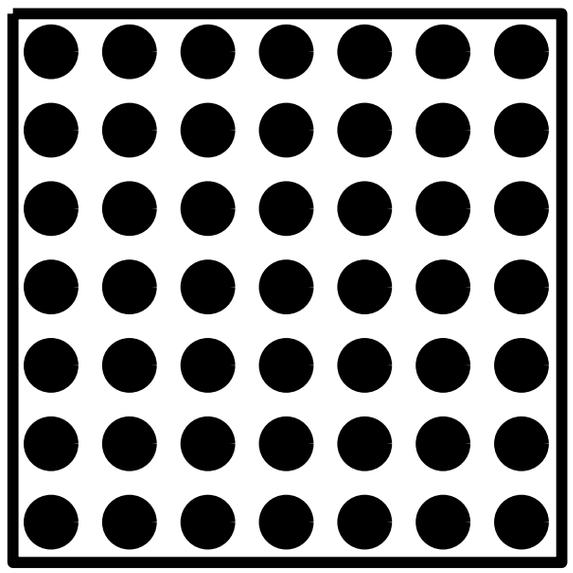
Discretizing (BVP) using a standard second order finite difference scheme, we get

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

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



Sparsity pattern of $\mathbf{A}$.



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

Inversion of structured matrices: Tridiagonal

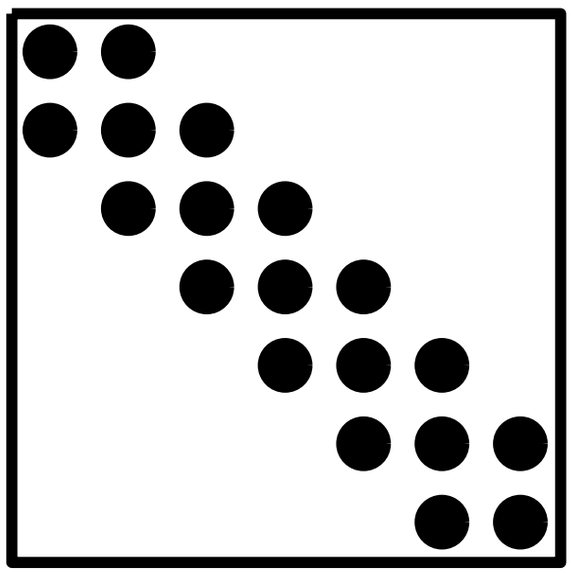
Consider a simple 2-point BVP on the interval $[0, 1]$:

$$\text{(BVP)} \qquad \begin{cases} -\dfrac{d^2u}{dx^2} + p(x)\dfrac{du(x)}{dx} + m(x)\,u(x) = g(x), & x \in (0,1), \\ u(0) = f_{\mathrm{L}}, \\ u(1) = f_{\mathrm{R}}. \end{cases}$$

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

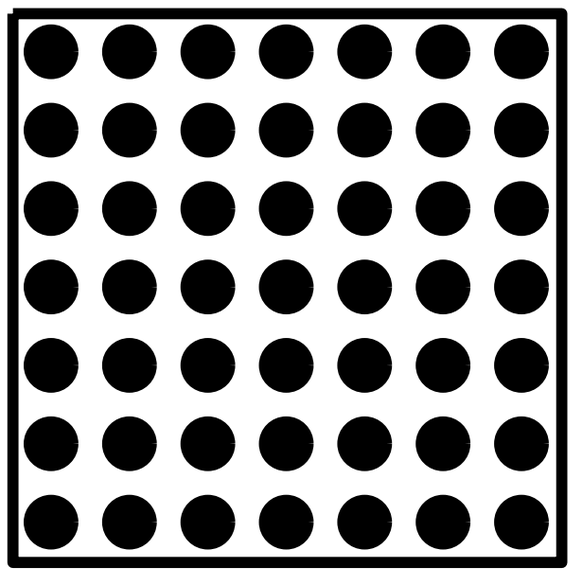
$$\mathbf{Au} = \mathbf{b},$$

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



Sparsity pattern of $\mathbf{A}$*.*

$\mathbf{A}$ *is tridiagonal.*



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

Inversion of structured matrices: Tridiagonal

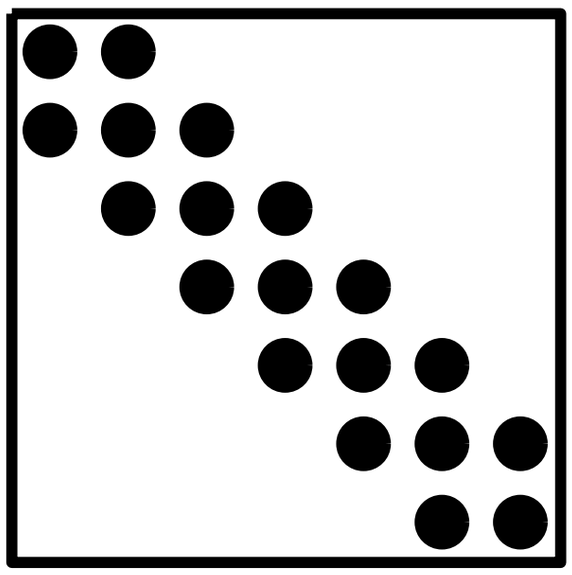
Consider a simple 2-point BVP on the interval $[0, 1]$:

$$\text{(BVP)} \qquad \begin{cases} -\dfrac{d^2u}{dx^2} + p(x)\dfrac{du(x)}{dx} + m(x)\,u(x) = g(x), & x \in (0,1), \\ u(0) = f_{\mathrm{L}}, \\ u(1) = f_{\mathrm{R}}. \end{cases}$$

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

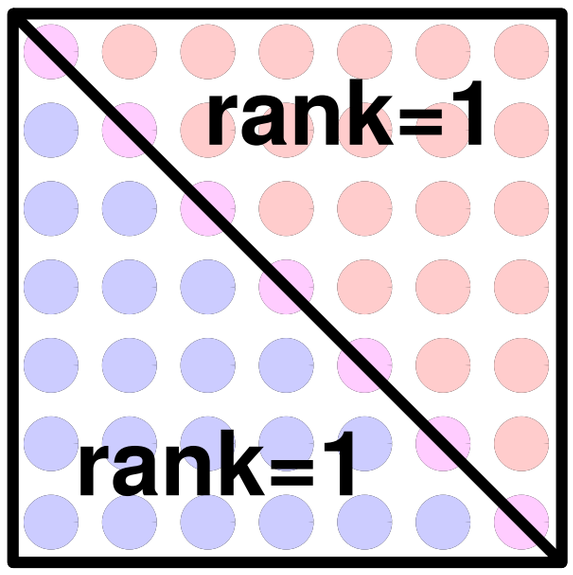
$$\mathbf{Au} = \mathbf{b},$$

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



Sparsity pattern of $\mathbf{A}$*.*

$\mathbf{A}$ *is tridiagonal.*



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

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

Inversion of structured matrices: Tridiagonal

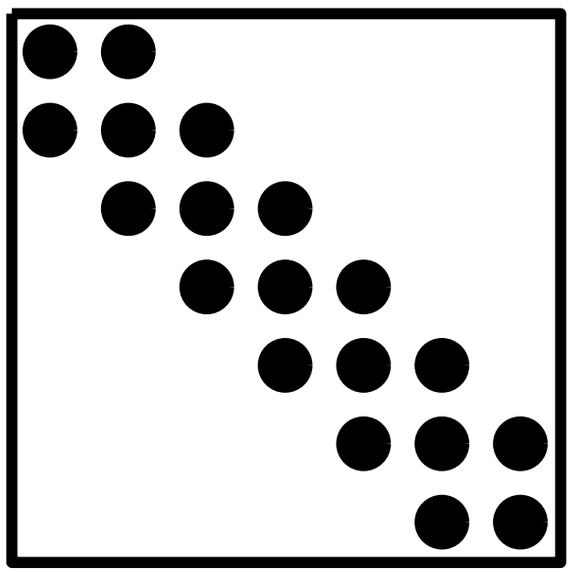
Consider a simple 2-point BVP on the interval $[0, 1]$:

$$\text{(BVP)} \qquad \begin{cases} -\dfrac{d^2u}{dx^2} + p(x)\dfrac{du(x)}{dx} + m(x)\,u(x) = g(x), & x \in (0,1), \\ u(0) = f_{\mathrm{L}}, \\ u(1) = f_{\mathrm{R}}. \end{cases}$$

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

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

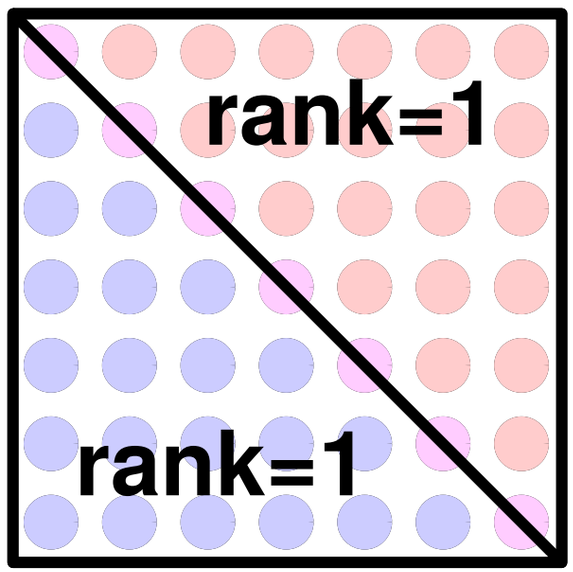
where $\mathbf{A}$ is a sparse matrix of size, say, $n \times n$. Then $\mathbf{A}^{-1}$ is dense.



Sparsity pattern of $\mathbf{A}$.

$\mathbf{A}$ *is tridiagonal.*

$\mathbf{A}$ *is sparse.*



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

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

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

Inversion of structured matrices: Tridiagonal

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

$$\text{(BVP)} \qquad \begin{cases} -\dfrac{d^2u}{dx^2} + p(x)\dfrac{du(x)}{dx} + m(x)\,u(x) = g(x), & x \in (0,1), \\ u(0) = f_{\mathrm{L}}, \\ u(1) = f_{\mathrm{R}}. \end{cases}$$

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

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

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

Fun facts:

- If $\mathbf{A}$ is invertible and tridiagonal, then $\mathbf{A}^{-1}$ is semi-separable (meaning that the upper triangular and the lower triangular parts are restrictions of rank 1 matrices).
- If $\mathbf{A}$ is invertible and semi-separable, then $\mathbf{A}^{-1}$ is tridiagonal.
- Cost of storage is $3n - 2$ floats in either case.
- Cost of inversion is $O(n)$ in either case.

(Note: LU factorization is more common: Factors $\mathbf{L}$ and $\mathbf{U}$ are each bidiagonal.)

Inversion of structured matrices: Semi-separable plus diagonal

Let us again consider a two point boundary value problem

$$\text{(BVP)} \qquad -u''(y) + m(y)\,u(y) = g(y), \qquad y \in (0,1),$$

now with zero boundary data. Recall that when $m = 0$, we can solve (BVP) analytically:

$$u(x) = \int_0^1 G(x,y)\,g(y)\,dy,$$

where the *Green's function* G (which is semi-separable!) takes the form

$$G(x,y) = \begin{cases} \dfrac{(b-x)(y-a)}{b-a}, & \text{when } x \geq y \quad \text{(on or below the diagonal)}, \\ \dfrac{(x-a)(b-y)}{b-a}, & \text{when } x \leq y \quad \text{(on or above the diagonal)}. \end{cases}$$

Multiply (BVP) by $G(x,y)$ and integrate in y over $[0,1]$ to get

$$u(x) + \int_0^1 G(x,y)\,m(y)\,u(y)\,dy = h(x), \qquad x \in [0,1],$$

where

$$h(x) = \int_0^1 G(x,y)\,g(y)\,dy.$$

Fact 1: The equation (BVP) is equivalent to

$$\text{(IE)} \qquad u(x) + \int_0^1 G(x,y)\,m(y)\,u(y)\,dy = h(x), \qquad x \in [0,1],$$

Inversion of structured matrices: Semi-separable plus diagonal

Let us again consider a two point boundary value problem

$$\text{(BVP)} \qquad -u''(y) + m(y)\,u(y) = g(y), \qquad y \in (0,1),$$

now with zero boundary data.

Fact 1: The equation (BVP) is equivalent to

$$\text{(IE)} \qquad u(x) + \int_0^1 G(x,y)\,m(y)\,u(y)\,dy = h(x), \qquad x \in [0,1],$$

where the *Green's function* (which is semi-separable!) takes the form

$$G(x,y) = \begin{cases} \dfrac{(b-x)(y-a)}{b-a}, & \text{when } x \geq y \quad \text{(on or below the diagonal)}, \\ \dfrac{(x-a)(b-y)}{b-a}, & \text{when } x \leq y \quad \text{(on or above the diagonal)}. \end{cases}$$

Inversion of structured matrices: Semi-separable plus diagonal

Let us again consider a two point boundary value problem

$$\text{(BVP)} \qquad -u''(y) + m(y)\,u(y) = g(y), \qquad y \in (0,1),$$

now with zero boundary data.

Fact 1: The equation (BVP) is equivalent to

$$\text{(IE)} \qquad u(x) + \int_0^1 G(x,y)\,m(y)\,u(y)\,dy = h(x), \qquad x \in [0,1],$$

where the *Green's function* (which is semi-separable!) takes the form

$$G(x,y) = \begin{cases} \dfrac{(b-x)(y-a)}{b-a}, & \text{when } x \geq y \quad \text{(on or below the diagonal)}, \\ \dfrac{(x-a)(b-y)}{b-a}, & \text{when } x \leq y \quad \text{(on or above the diagonal)}. \end{cases}$$

Fact 2: Discretizing (IE) using a Nyström method with a uniform grid $\{x_i\}_{i=0}^{n+1} \subset [0,1]$ and the basic Trapezoidal rule results in the linear system

$$\big(\mathbf{I} + \mathbf{GM}\big)\mathbf{u} = \mathbf{Gg}.$$

The $n \times n$ matrix $\mathbf{G}$ is *semi-separable* since it has entries

$$\mathbf{G}(i,j) = h\,G(x_i, x_j).$$

The matrix $\mathbf{M}$ is the diagonal matrix whose diagonal entries are $\{m(x_i)\}_{i=1}^{n}$.

Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization $\rightarrow$ *sparse* linear system.
- Integral equation (IE) discretization $\rightarrow$ *dense* linear system.

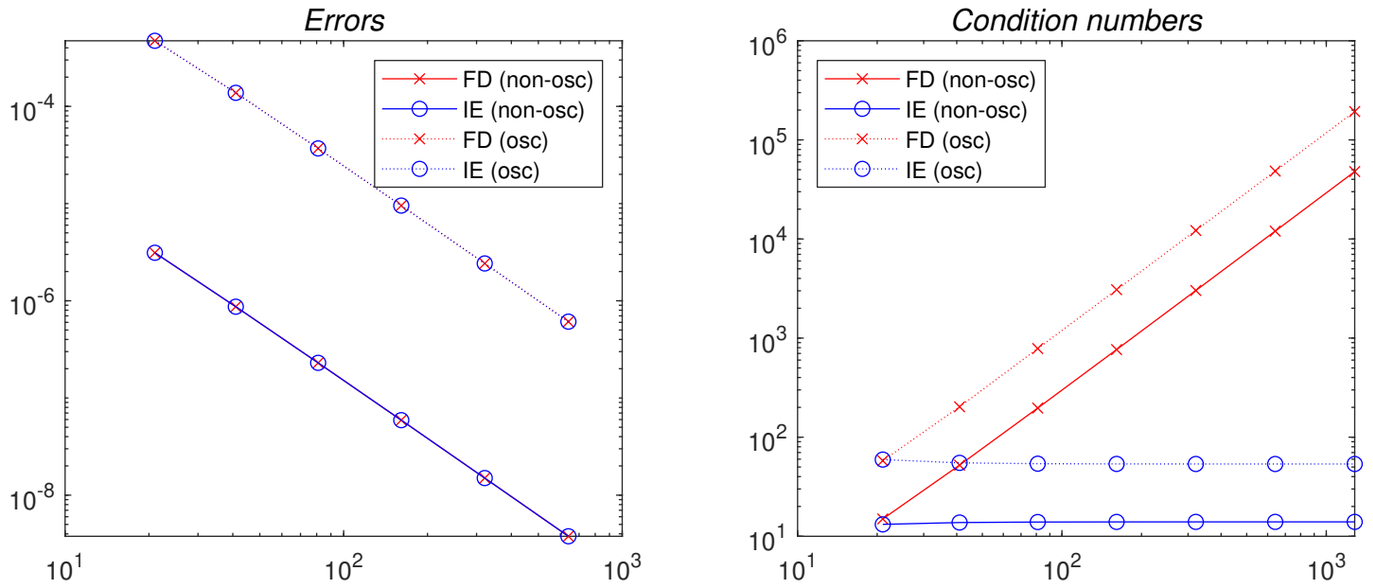
How do the two methods compare?

Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization $\rightarrow$ *sparse* linear system.
- Integral equation (IE) discretization $\rightarrow$ *dense* linear system.

How do the two methods compare?



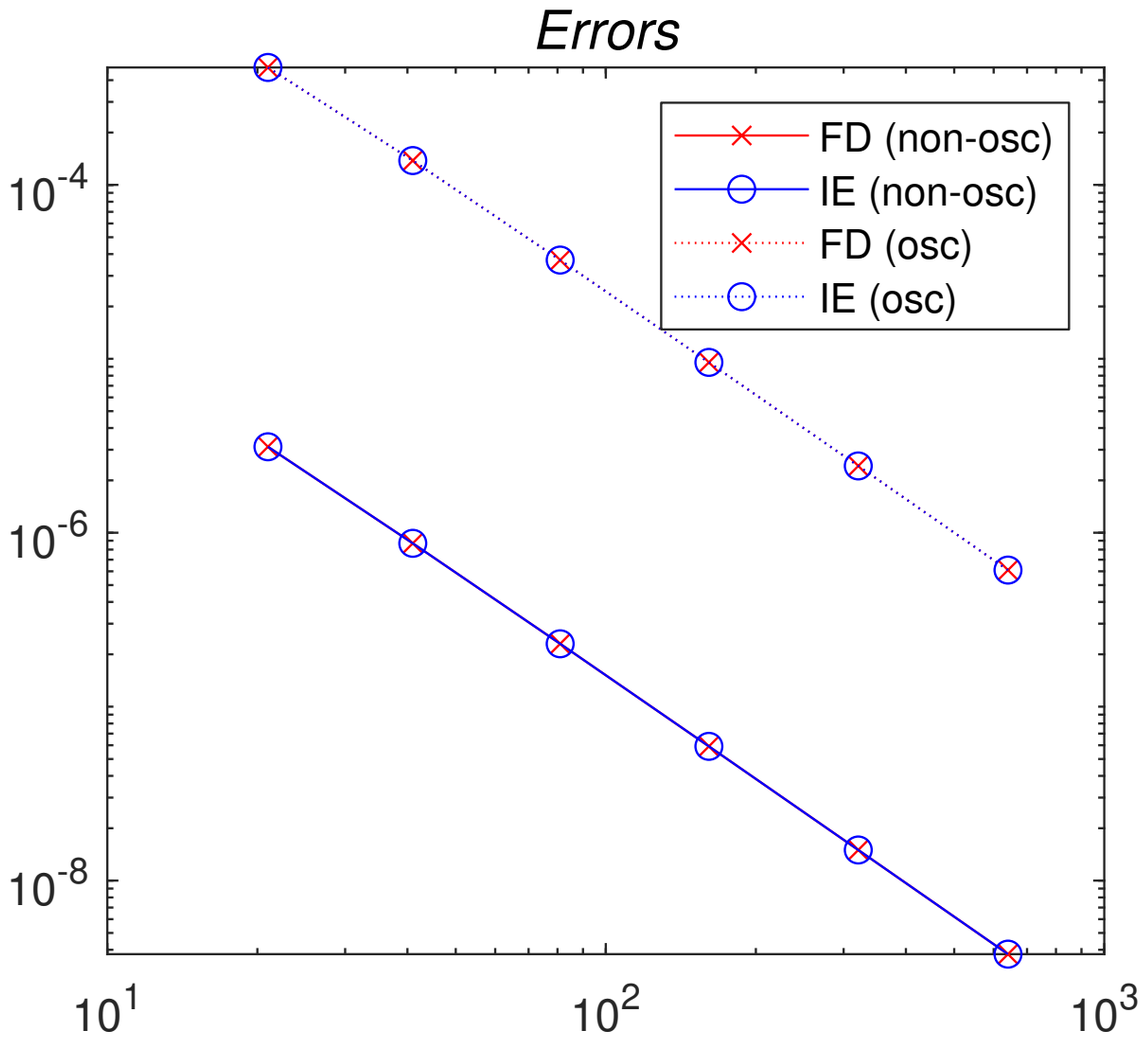
We considered a non-oscillatory problem "(non-osc)" where $m(x) = 100(1+x)\cos(x)$ and $g(x) = 1 + \cos(1+x)$. We then swapped the sign of m (but kept everything else the same) to get a problem with an oscillatory solution "(osc)". The left plot shows the errors incurred (in max norm), while the right one shows the condition numbers of the coefficient matrices. The key point here is that the condition numbers of the integral equation formulation does not grow with n. A secondary point is to show that elliptic problems with oscillatory solutions are far more challenging.

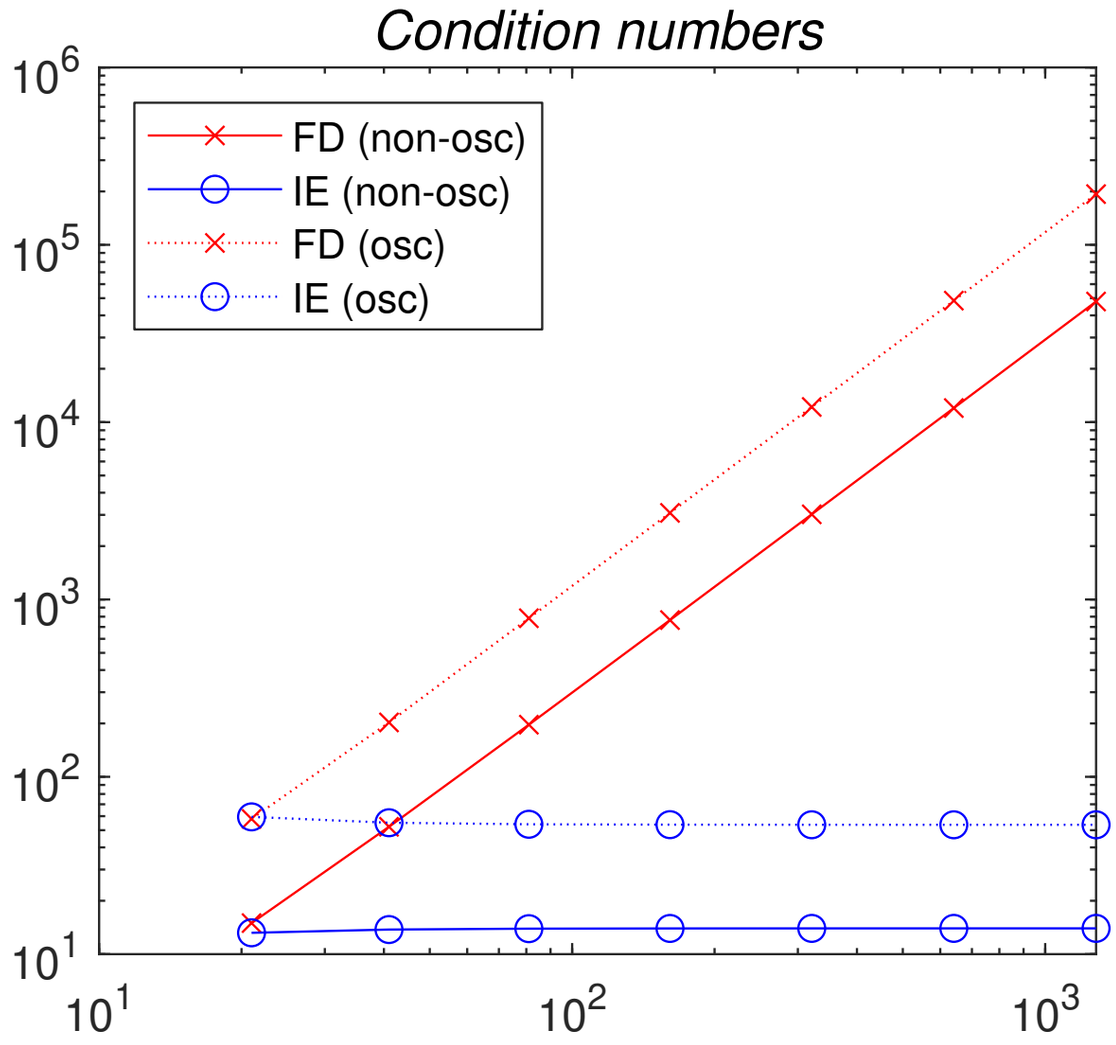
Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization → *sparse* linear system.
- Integral equation (IE) discretization → *dense* linear system.

How do the two methods compare?





Note: *For the IE, the condition number is small and constant!*

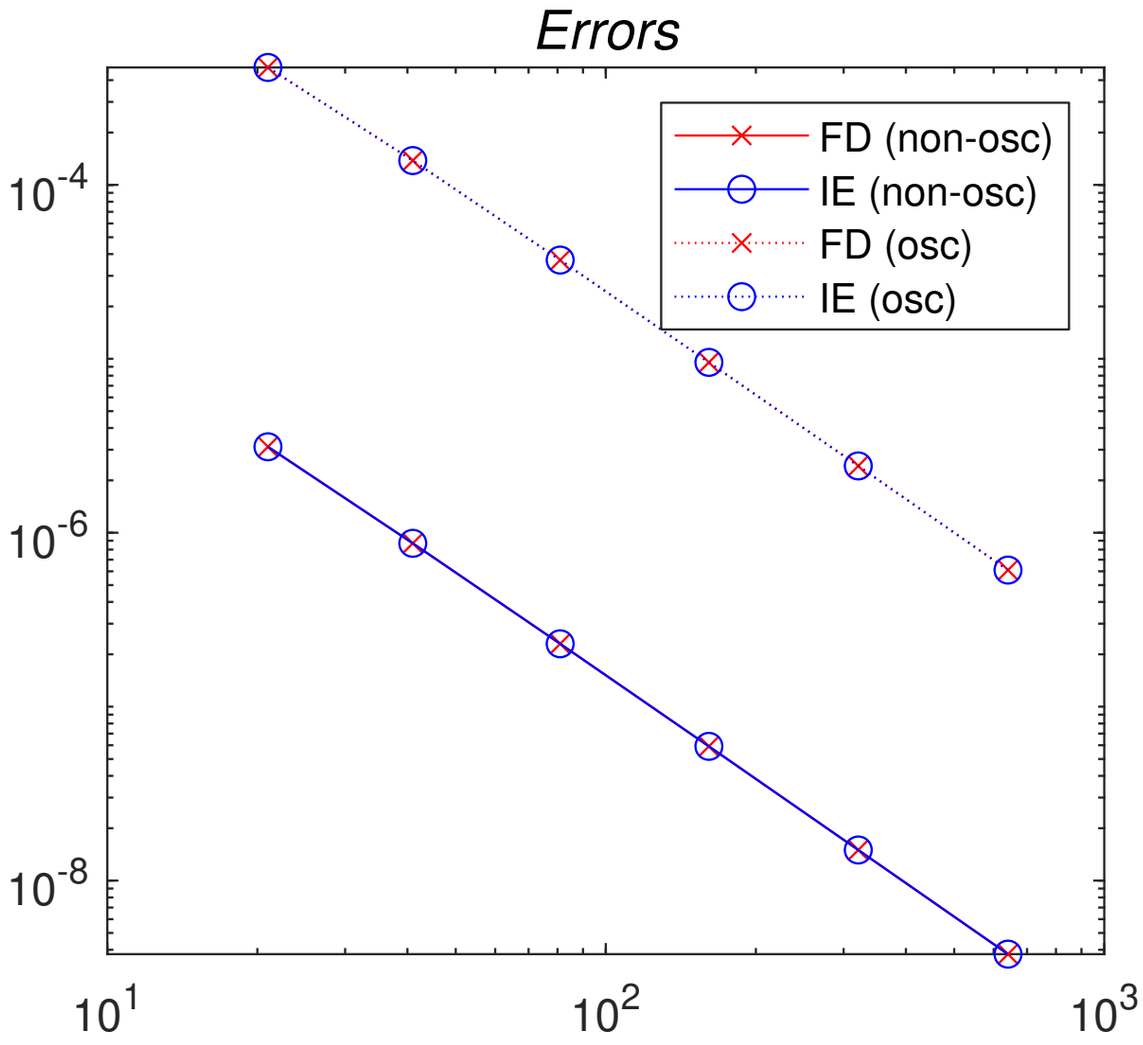
(Fun fact: The two methods are mathematically equivalent – errors are *identical!*)

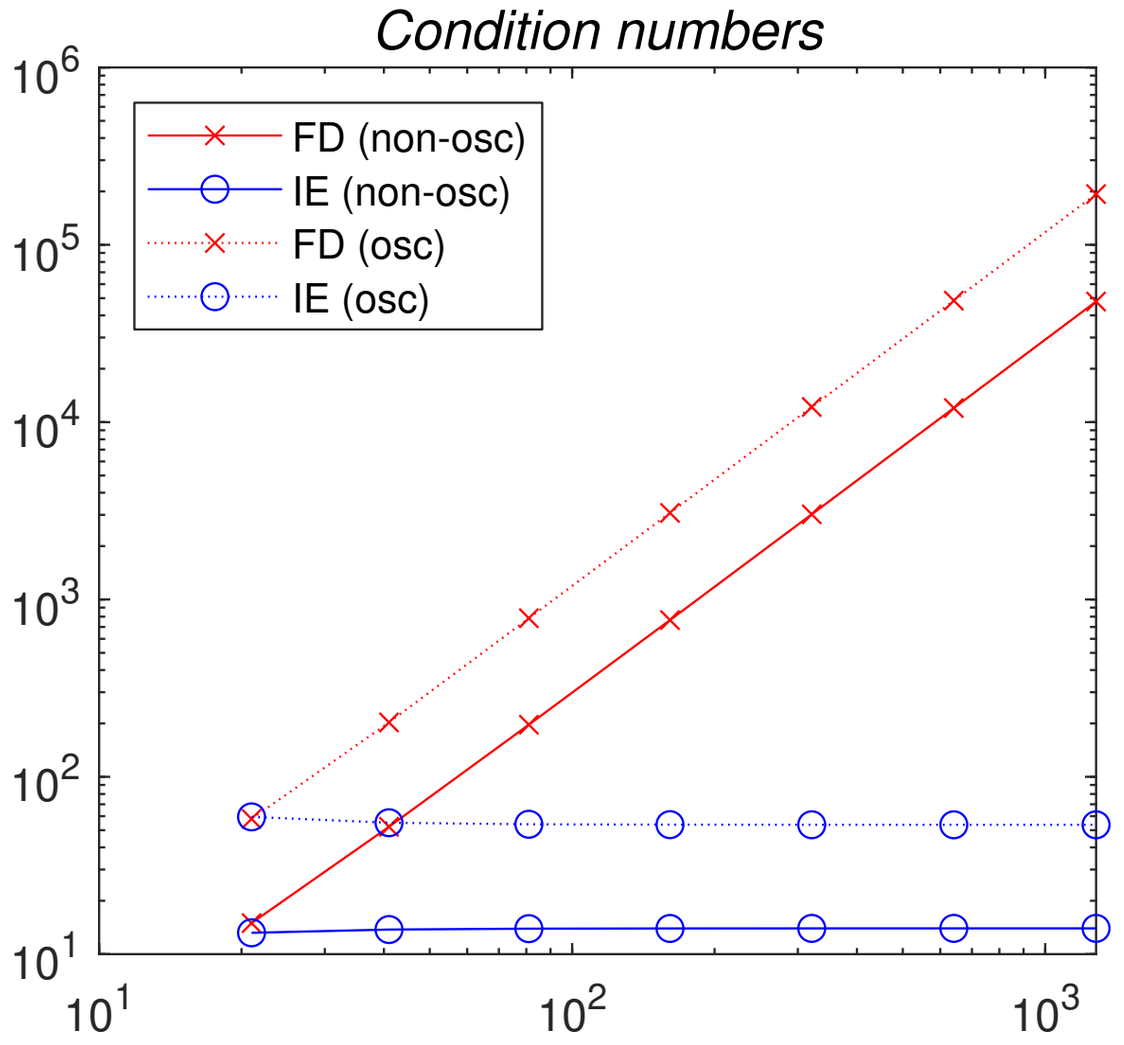
Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization → *sparse* linear system.
- Integral equation (IE) discretization → *dense* linear system.

How do the two methods compare?





The conversion to an IE is an example of “analytic preconditioning”.

You do the preconditioning mathematically, before you discretize.

Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization $\rightarrow$ *sparse* linear system.
- Integral equation (IE) discretization $\rightarrow$ *dense* linear system.

How do the two methods compare?

What about computational costs?

Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization $\rightarrow$ *sparse* linear system.
- Integral equation (IE) discretization $\rightarrow$ *dense* linear system.

How do the two methods compare?

What about computational costs?

Cost of computing an inverse is $O(n)$ in either case!

Inversion of structured matrices: Semi-separable plus diagonal

Observe that we introduced two different approaches for solving 2 point BVPs:

- Finite difference (FD) discretization $\rightarrow$ *sparse* linear system.
- Integral equation (IE) discretization $\rightarrow$ *dense* linear system.

How do the two methods compare?

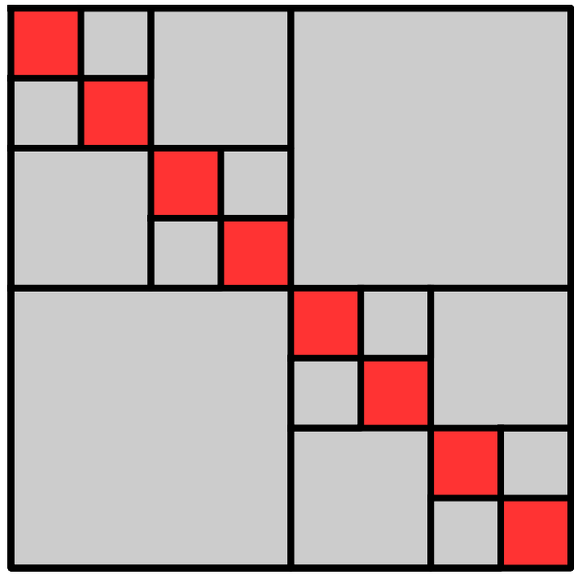
What about computational costs?

Cost of computing an inverse is $O(n)$ in either case!

We skip details for the "semi-separable plus diagonal" case, and instead go straight to a more general class: HODLR.

Inversion of structured matrices: HODLR

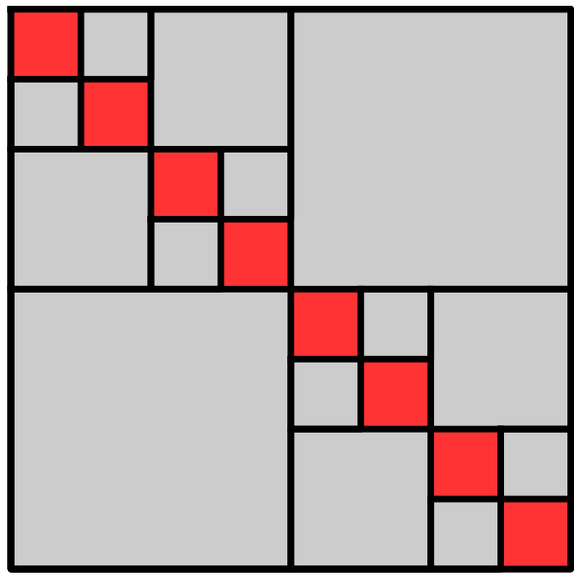
Now let us consider a slightly more complex structure:



All gray blocks have low rank.

Inversion of structured matrices: HODLR

Now let us consider a slightly more complex structure:



All gray blocks have low rank.

Let us start with a simple *recursive* inversion procedure.

The first step is to observe that if we tessellate $\mathbf{A}$ as follows

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

then

- $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ each have low numerical rank,
- $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ each are HODLR themselves.

Note: *"Semi-separable + diagonal" is a special case of HODLR.*

Inversion of structured matrices: HODLR

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

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

Inversion of structured matrices: HODLR

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

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

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

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

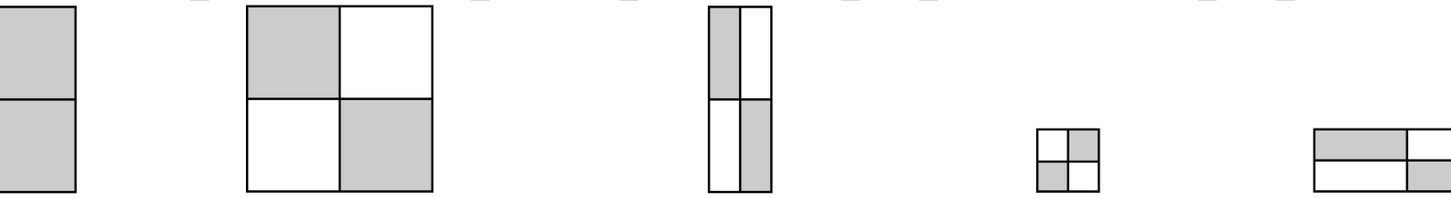
Inversion of structured matrices: HODLR

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

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

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

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



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

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

Inversion of structured matrices: HODLR

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

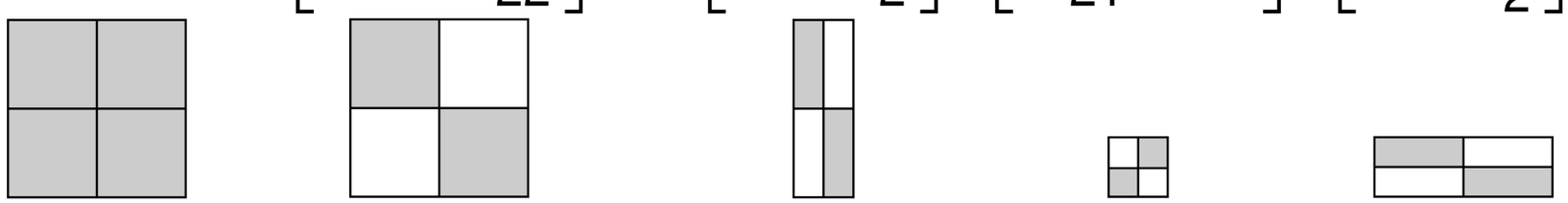
$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

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

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

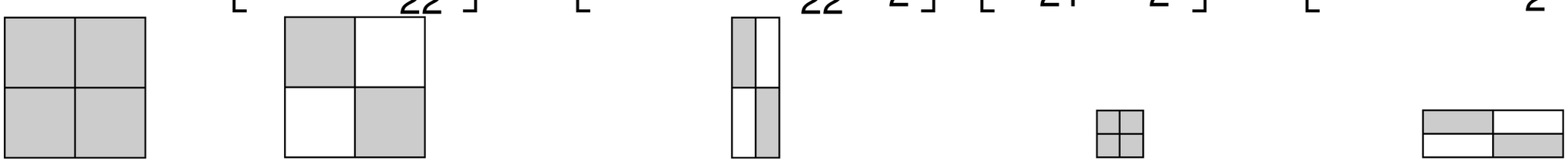
Then we can write $\mathbf{A}$ in the form

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



Applying the Woodbury formula, we get

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



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

Inversion of structured matrices: HODLR

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

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

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

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

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

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

Applying the Woodbury formula, we get

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

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

Inversion of structured matrices: HODLR

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

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

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

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

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

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

Applying the Woodbury formula, we get

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

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

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

Obvious recursion!

Inversion of structured matrices: HODLR

The recursive inversion formula for a HODLR matrix is conceptually simple.

But it is hard to code efficiently, and leads to growth of the numerical ranks.

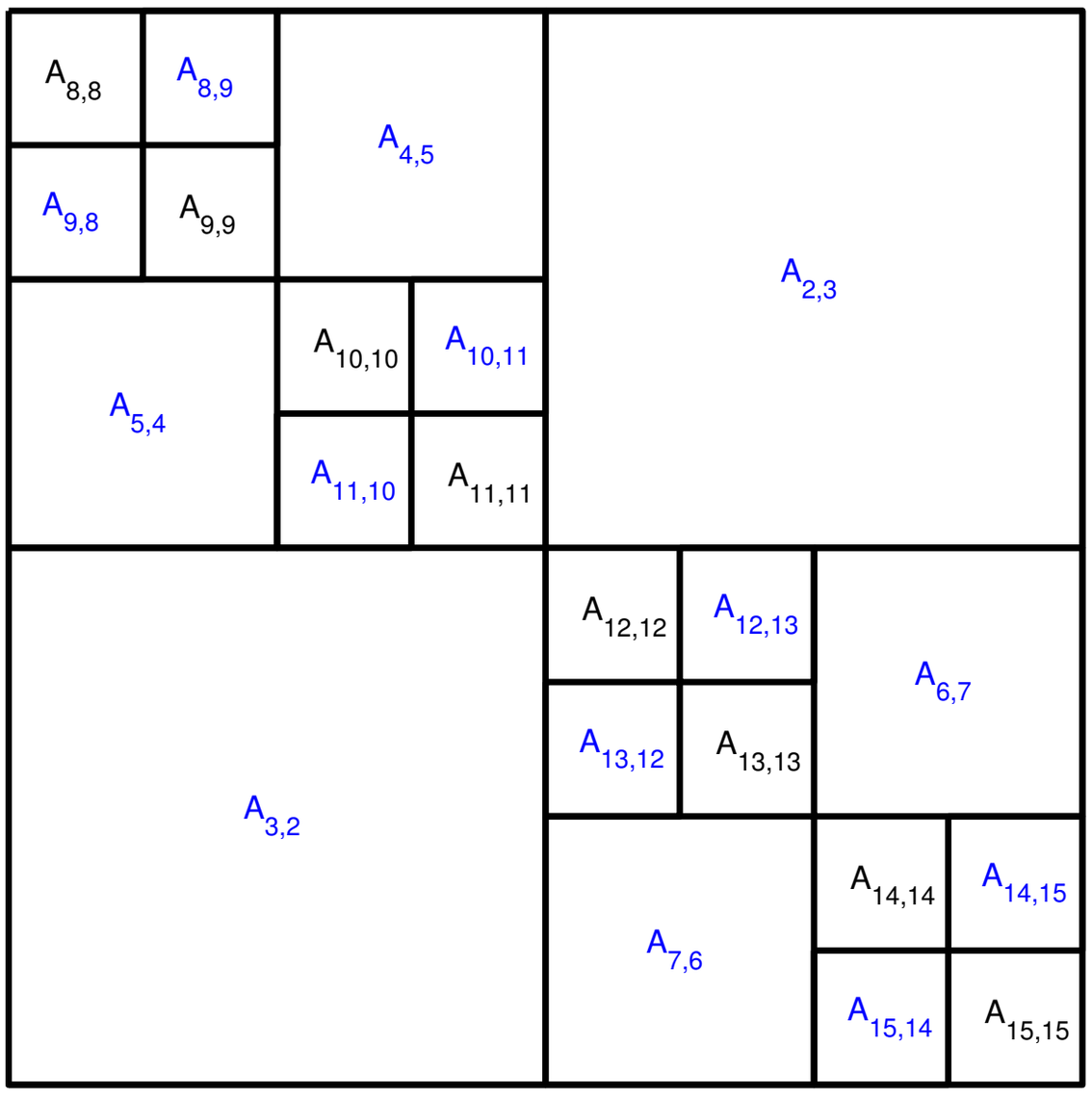
Inversion of structured matrices: HODLR

The recursive inversion formula for a HODLR matrix is conceptually simple.

But it is hard to code efficiently, and leads to growth of the numerical ranks.

Luckily, there are better options, including non-recursive *exact* formulas.

Let us consider a specific example:



For every sibling pair $\{\alpha, \beta\}$ set

$$\mathbf{A}_{\alpha,\beta} = \mathbf{A}(I_\alpha, I_\beta).$$

Fix a bound k on the rank, and a tolerance ε. We then require that each off-diagonal block (in blue in the figure) have ε-rank at most k. Define factors

$$\underset{n_\alpha \times n_\beta}{\mathbf{A}_{\alpha,\beta}} = \underset{n_\alpha \times k}{\mathbf{U}_\alpha} \; \underset{k \times n_\beta}{\mathbf{V}_\beta^*}.$$

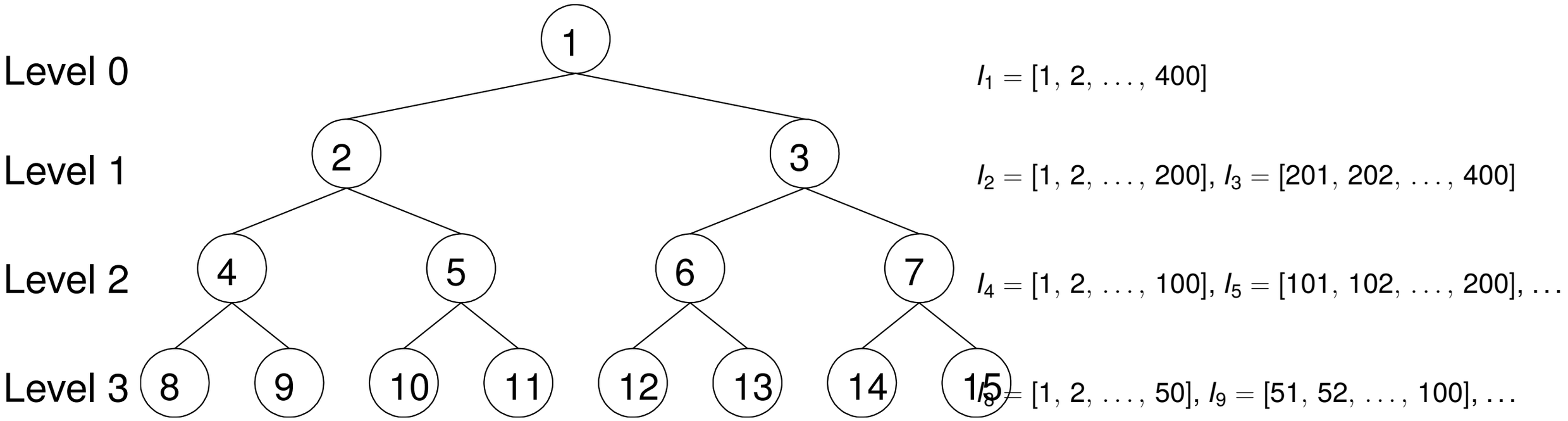
Inversion of structured matrices: HODLR

To more formally define the HODLR format, we need to introduce a *tree structure* on the index set $I = [1, 2, 3, \ldots, N]$.

Let $\mathbf{A}$ be an $N \times N$ matrix.

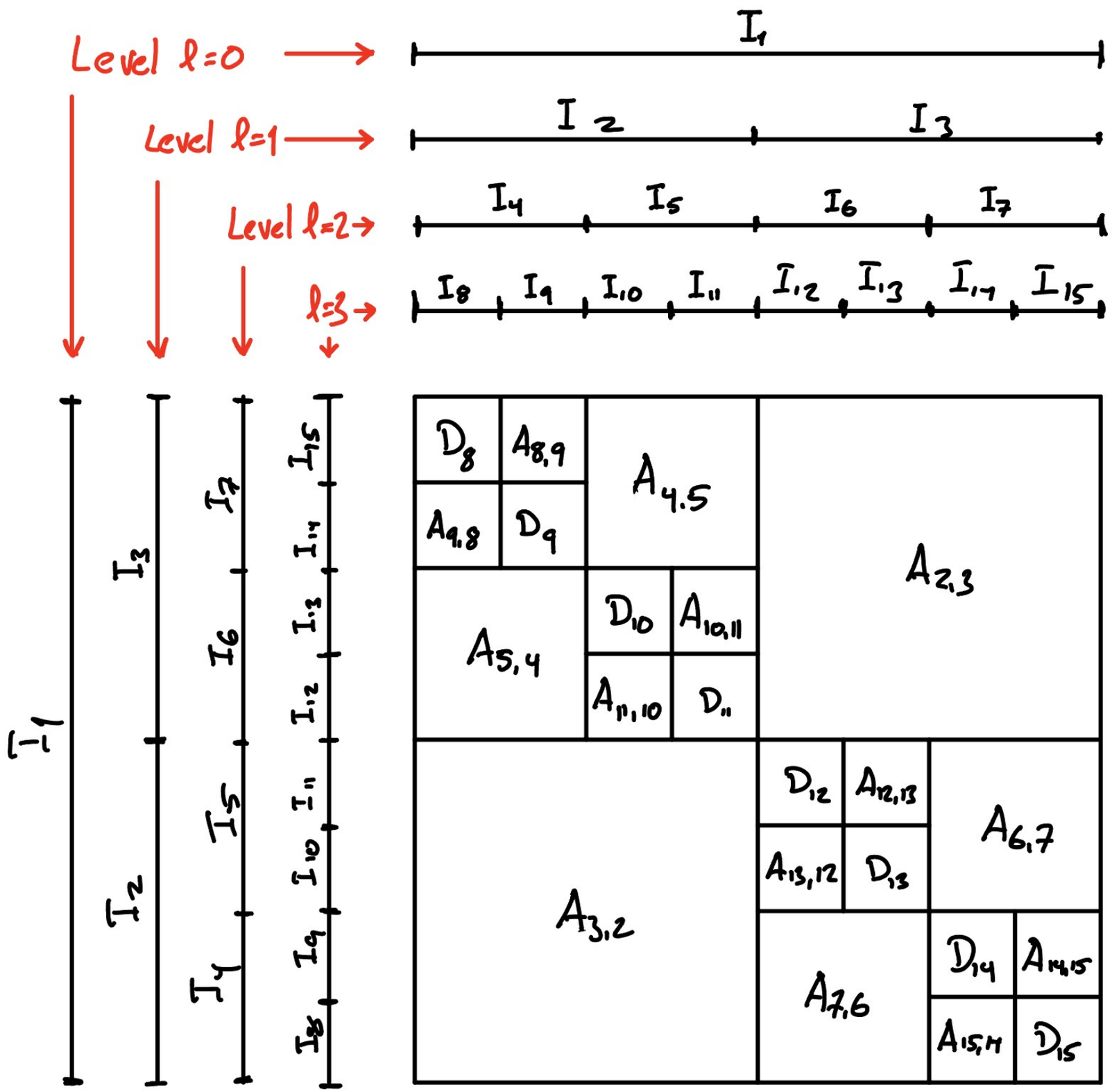
Suppose $\mathcal{T}$ is a binary tree on the index vector $I = [1, 2, 3, \ldots, N]$.

For a node τ in the tree, let I_τ denote the corresponding index vector.



For nodes σ and τ on the same level, set $\mathbf{A}_{\sigma,\tau} = \mathbf{A}(I_\sigma, I_\tau)$.

Inversion of structured matrices: HODLR



Inversion of structured matrices: HODLR

For our 3-level model problem, we will build $\mathbf{A}^{-1}$ in the form

$$\mathbf{A}^{-1} = \mathbf{B}_0 \, \mathbf{B}_1 \, \mathbf{B}_2 \, \mathbf{B}_3,$$

where each $\mathbf{B}_\ell$ is block diagonal, with diagonal blocks that are rank-k perturbations of the identity matrix. Consequently, each $\mathbf{B}_\ell$ can be applied to a vector in $O(Nk)$ flops.

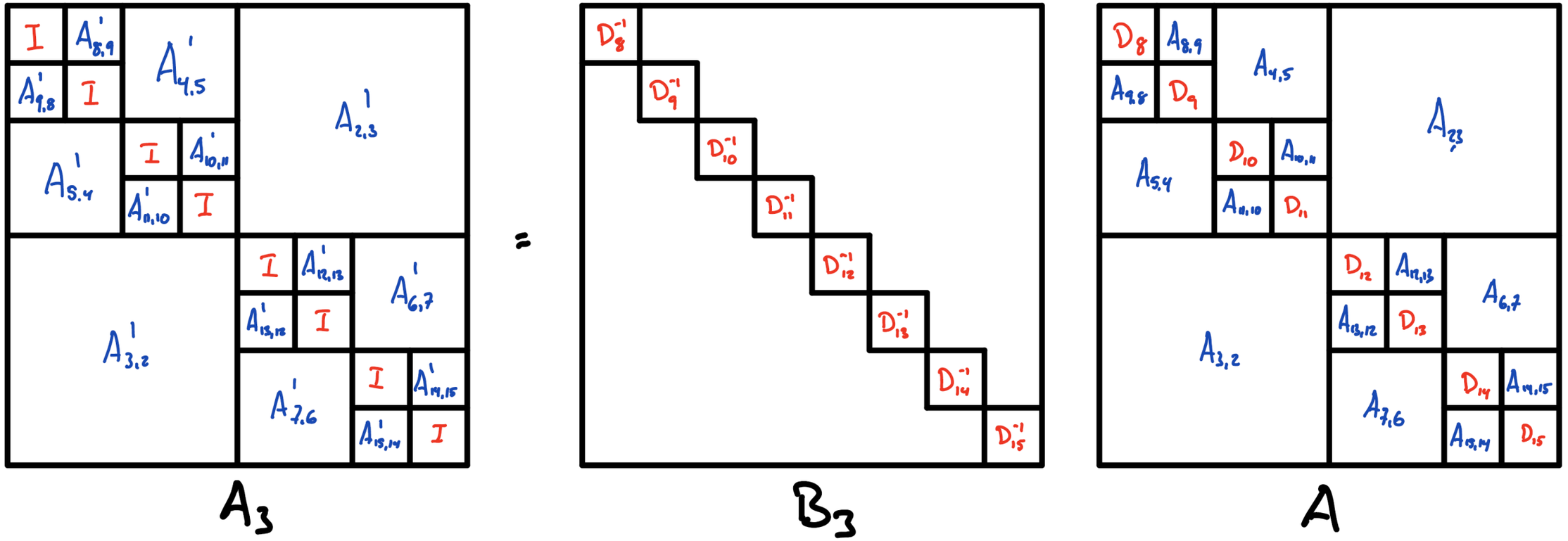
Inversion of structured matrices: HODLR

For our 3-level model problem, we will build $\mathbf{A}^{-1}$ in the form

$$\mathbf{A}^{-1} = \mathbf{B}_0\,\mathbf{B}_1\,\mathbf{B}_2\,\mathbf{B}_3,$$

where each $\mathbf{B}_\ell$ is block diagonal, with diagonal blocks that are rank-k perturbations of the identity matrix. Consequently, each $\mathbf{B}_\ell$ can be applied to a vector in $O(Nk)$ flops.

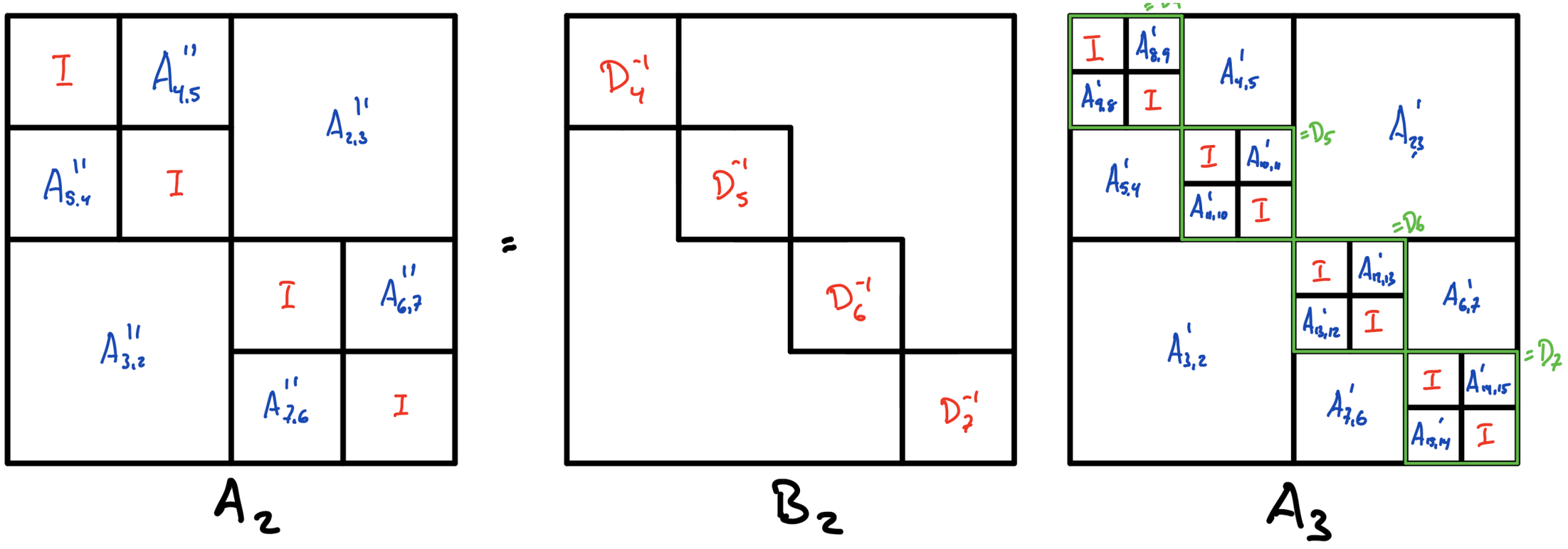
In the first step, we form a block diagonal matrix $\mathbf{B}_3$ whose diagonal blocks are the inverses of the diagonal blocks of $\mathbf{A}$. We then apply $\mathbf{B}_3$ to $\mathbf{A}$, to form a matrix $\mathbf{A}_3 = \mathbf{B}_3\mathbf{A}$ whose diagonal blocks are the identity matrix:



Observe that all the off-diagonal blocks in $\mathbf{A}_3$ still have rank at most k.

Inversion of structured matrices: HODLR

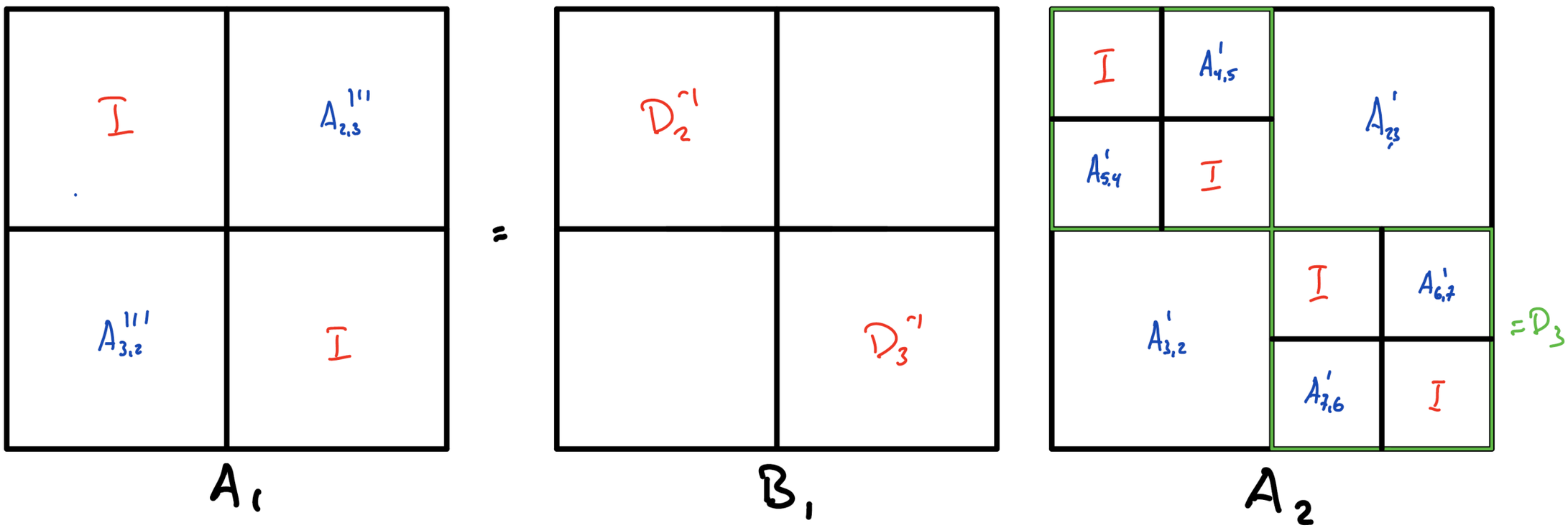
In the second step, let $\mathbf{D}_4$, $\mathbf{D}_5$, $\mathbf{D}_6$, $\mathbf{D}_7$ denote the diagonal blocks of $\mathbf{A}_3$, as marked in the figure below. Since each of these matrices are of the form "identity plus low rank", they can inexpensively be inverted. We put the inverses $\mathbf{D}_4^{-1}$, $\mathbf{D}_5^{-1}$, $\mathbf{D}_6^{-1}$, $\mathbf{D}_7^{-1}$ into the diagonal blocks of $\mathbf{B}_2$ and apply it to $\mathbf{A}_3$ to obtain



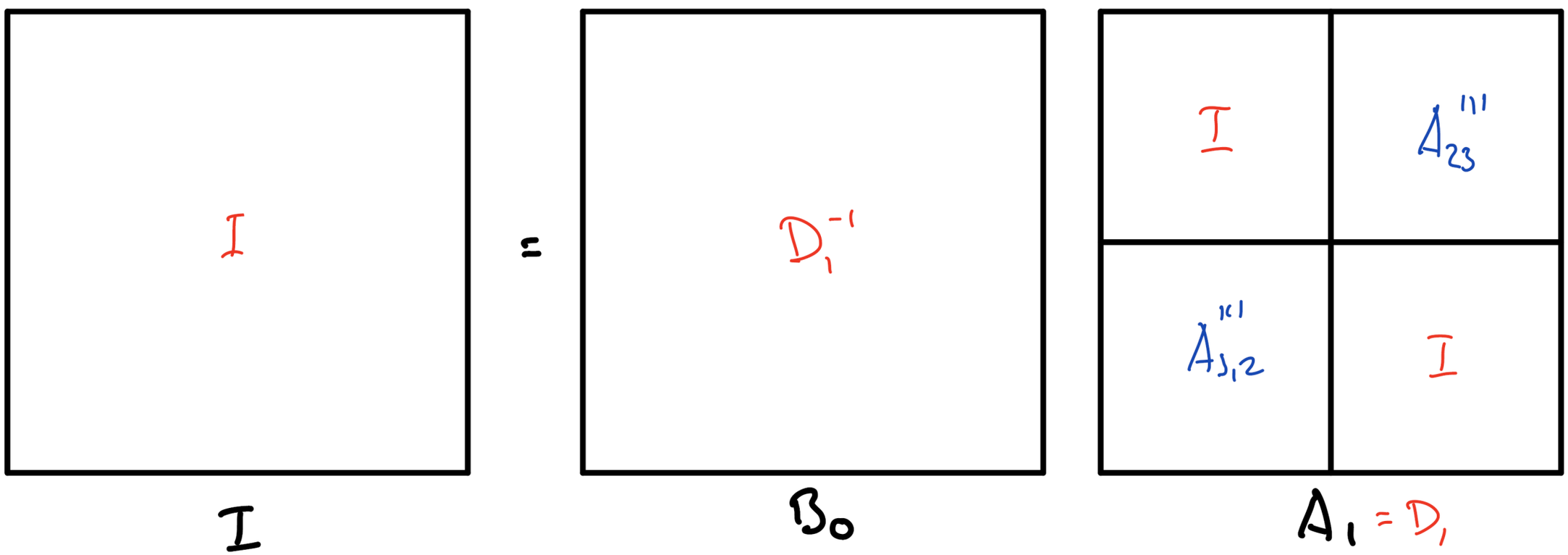
Observe again that the off-diagonal blocks of $\mathbf{A}_2$ all have rank at most k.

Inversion of structured matrices: HODLR

The third step:



The fourth and final step:

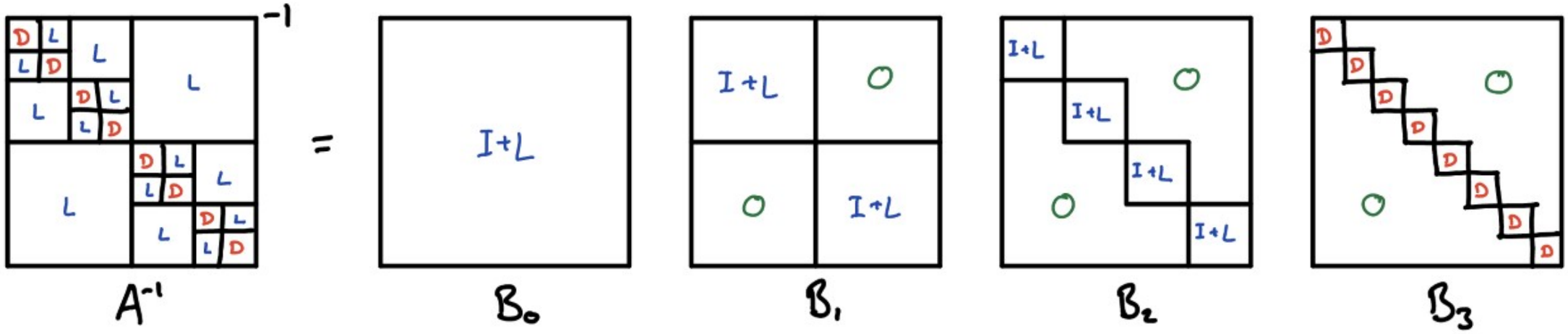


Inversion of structured matrices: HODLR

Once the process completes, we have obtained the factorization

$$\mathbf{I} = \mathbf{B}_0\mathbf{A}_0 = \mathbf{B}_0\mathbf{B}_1\mathbf{A}_1 = \mathbf{B}_0\mathbf{B}_1\mathbf{B}_2\mathbf{A}_2 = \mathbf{B}_0\mathbf{B}_1\mathbf{B}_2\mathbf{B}_3\mathbf{A}.$$

In other words,



where

"L" means "low rank"

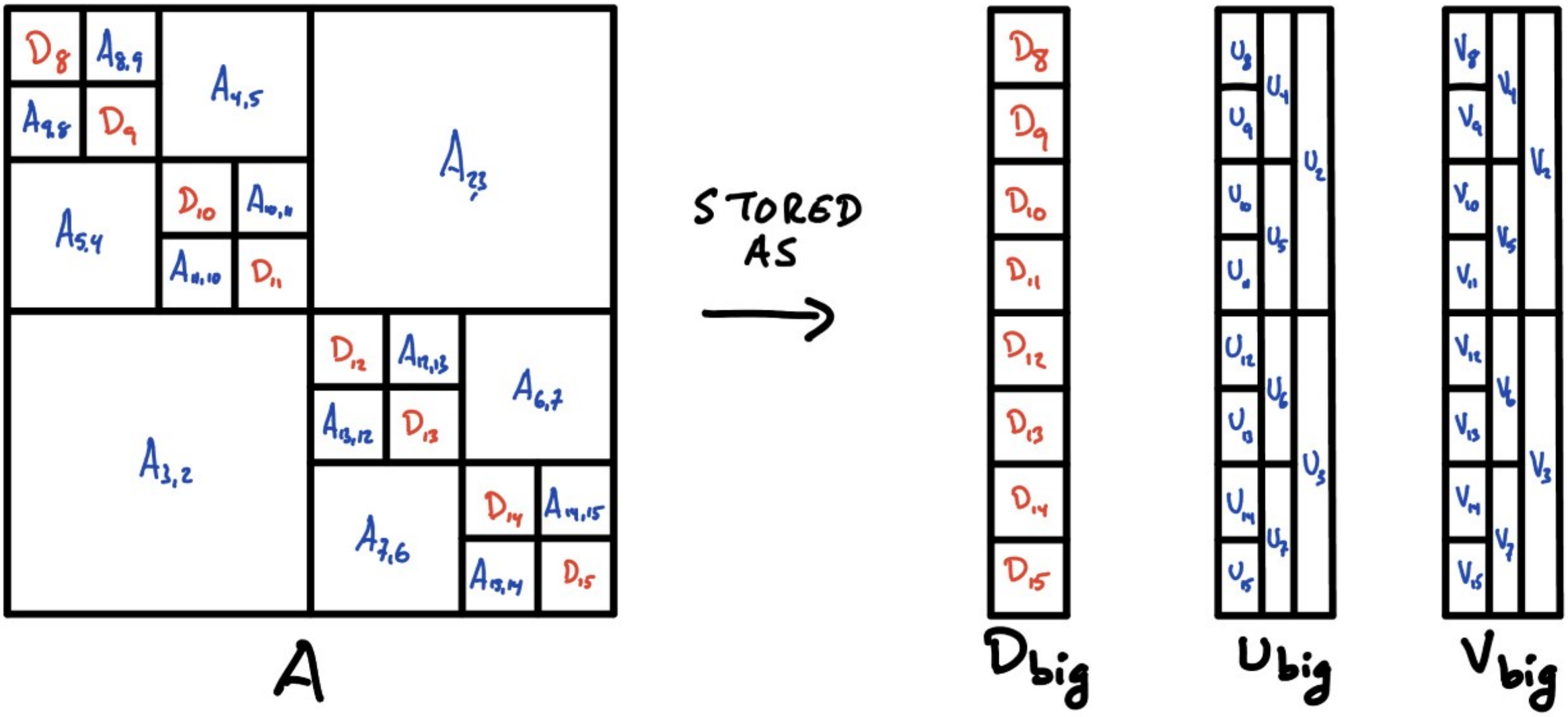
"D" means "dense"

All updates are ***multiplicative****.*

Inversion of structured matrices: HODLR

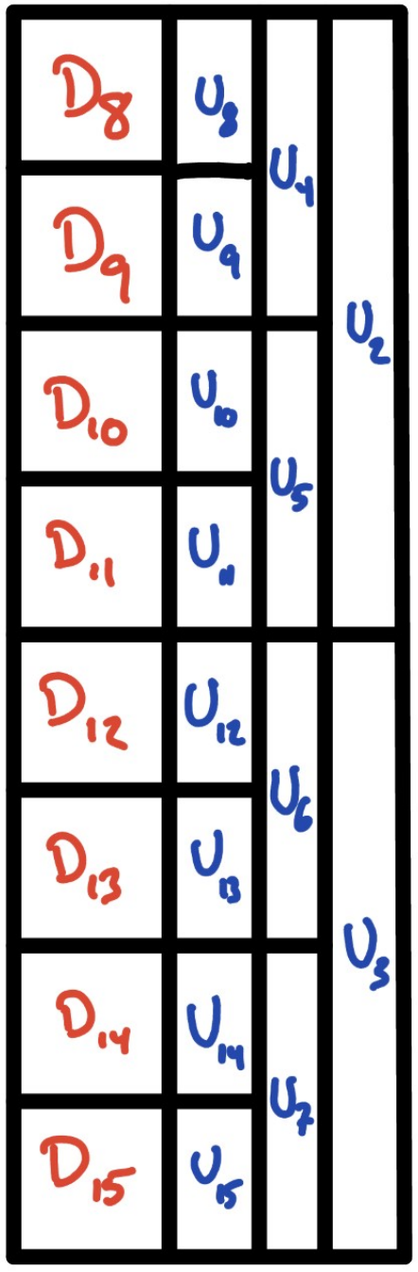
For purposes of practical execution speed, it is convenient to aggregate all the factors in a HODLR representation into three large arrays.

For a 3-level HODLR matrix, we do (assuming all ranks on a single level are identical):



The factorization procedure we described acts directly on the matrix $[\mathbf{D}_{\text{big}}\ \mathbf{U}_{\text{big}}]$, overwriting the original factors. All operations can be executed in *batched form*.

Inversion of structured matrices: HODLR



Inversion of structured matrices: Key points

- Dense matrices that contain low-rank structure often admit smart methods for storage, for matrix-vector multiplication, for inversion and LU factorization, etc.
- Linear, or close to linear, complexity is often attainable.
- There are many different “formats” for such matrices.
  For each format, there are many different algorithms for doing the same task!

Will discuss different formats in more depth in second session.

Sparse solvers

Linear complexity sparse direct solvers

Consider the problem of solving

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

where **A** is a sparse matrix resulting from discretization (FEM/FD/...) of a PDE.

Rank structured matrix algebra could in principle be used directly to build $\mathbf{A}^{-1}$.

However, it is more efficient to exploit the sparsity that is present.

Linear complexity sparse direct solvers

Consider the problem of solving

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

where $\mathbf{A}$ is a sparse matrix resulting from discretization (FEM/FD/...) of a PDE.

Rank structured matrix algebra could in principle be used directly to build $\mathbf{A}^{-1}$.

However, it is more efficient to exploit the sparsity that is present.

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.

Linear complexity sparse direct solvers

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

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

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

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

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

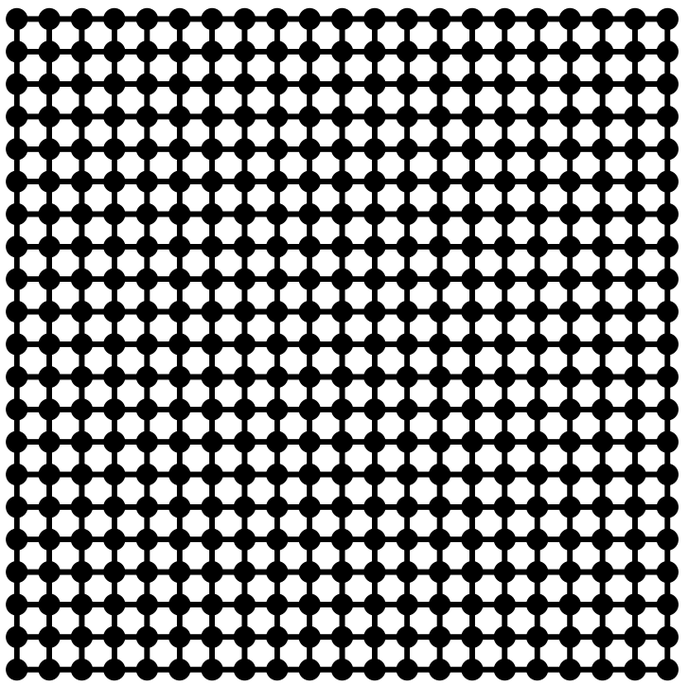


Figure A: The grid

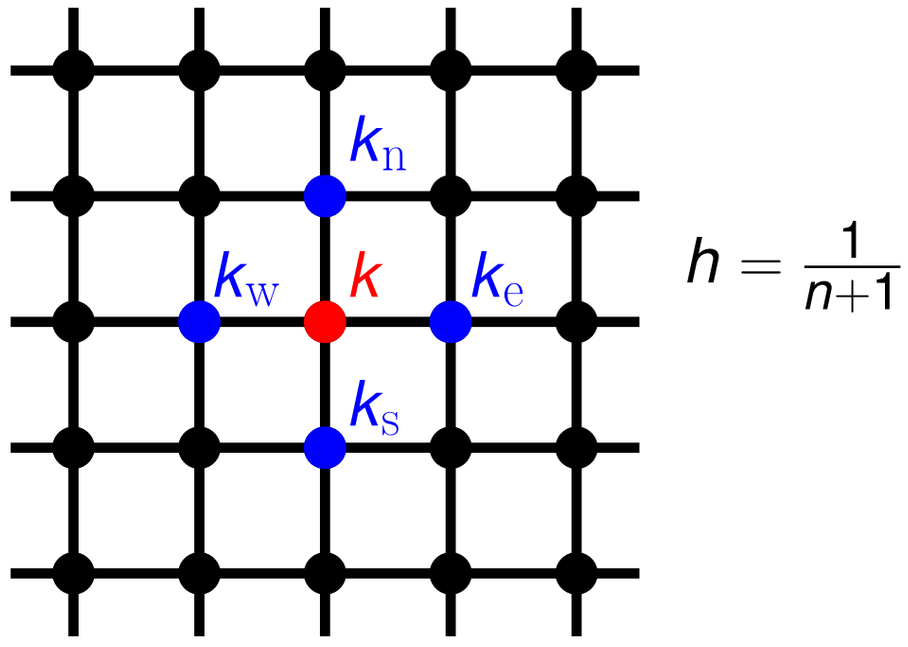
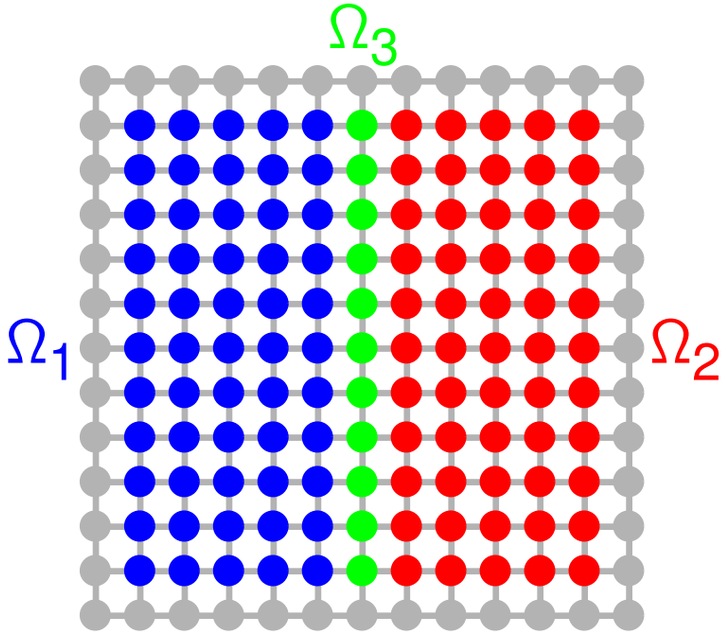


Figure B: The 5-point stencil

Linear complexity sparse direct solvers

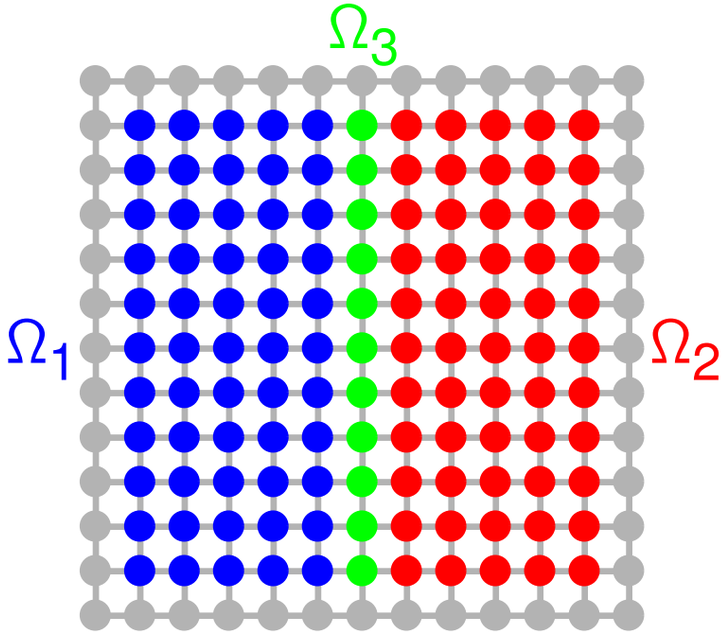
Divide-and-conquer: Split the nodes in three groups as shown so that there are no connections between nodes in Ω_1 and Ω_2. Then $\mathbf{A}$ has zero blocks as shown:



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

Linear complexity sparse direct solvers

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 **A** has zero blocks as shown:



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

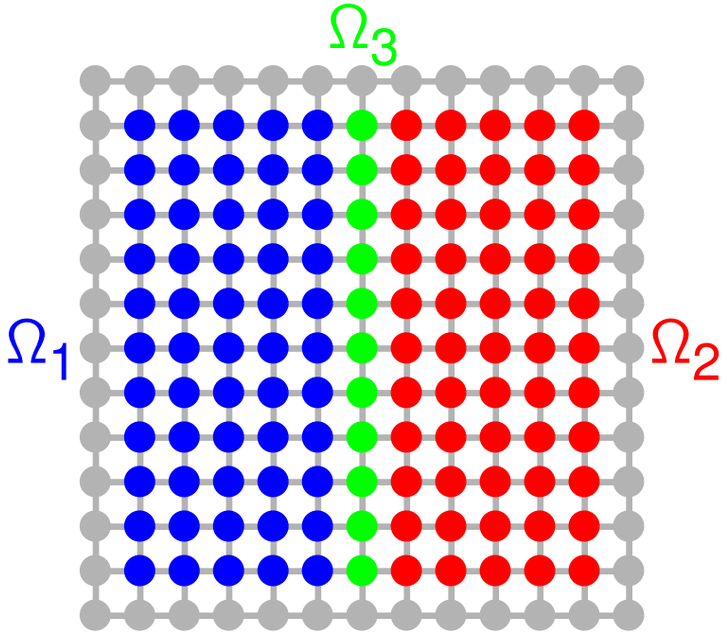
Now suppose that we can somehow construct $\mathbf{A}_{11}^{-1}$ and $\mathbf{A}_{22}^{-1}$. Then

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

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.*

Linear complexity sparse direct solvers

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



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

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

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

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

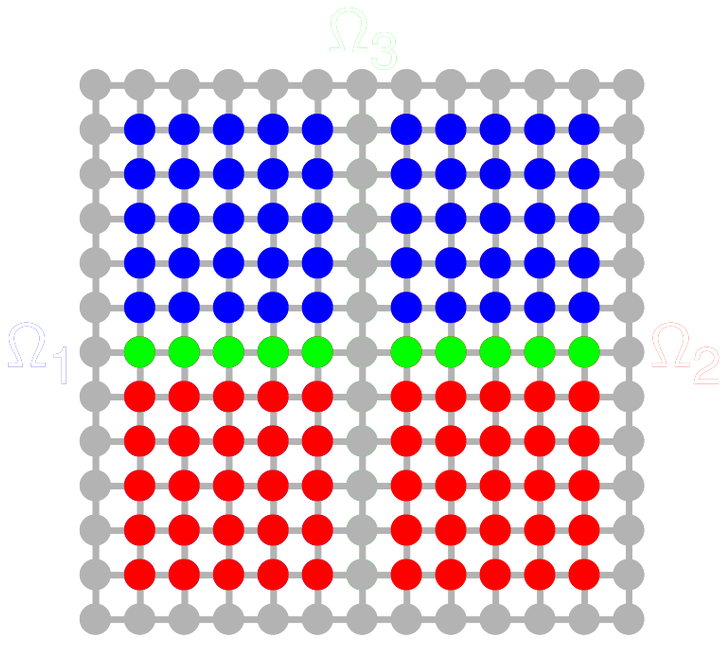
In other words, in order to invert $\mathbf{A}$, we need to execute three steps:

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

Notice the obvious recursion!

Linear complexity sparse direct solvers

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



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

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.*

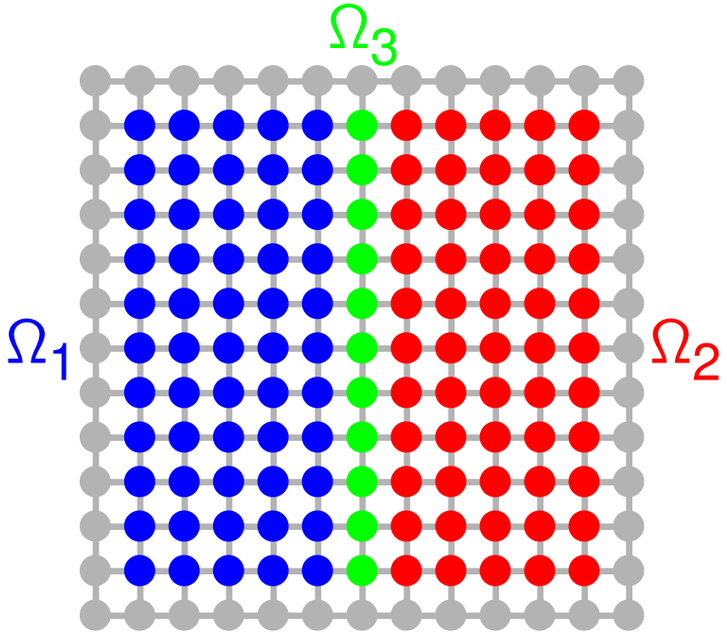
In other words, in order to invert $\mathbf{A}$, we need to execute three steps:

- 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!

Linear complexity sparse direct solvers

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



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

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

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

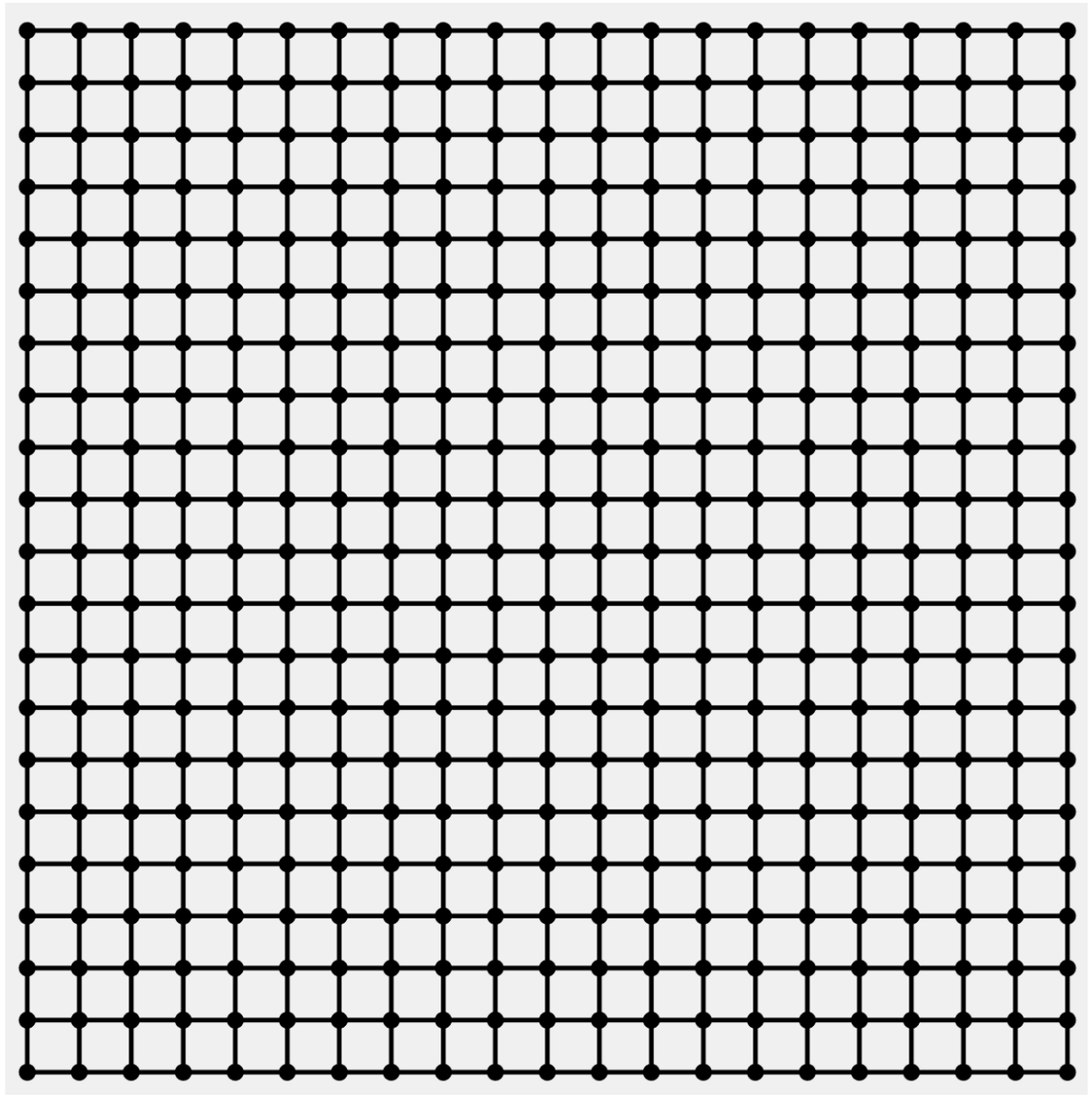
where $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{U}_{22}^{-1}\mathbf{L}_{22}^{-1}\mathbf{A}_{23}$ is a *Schur complement.*

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

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

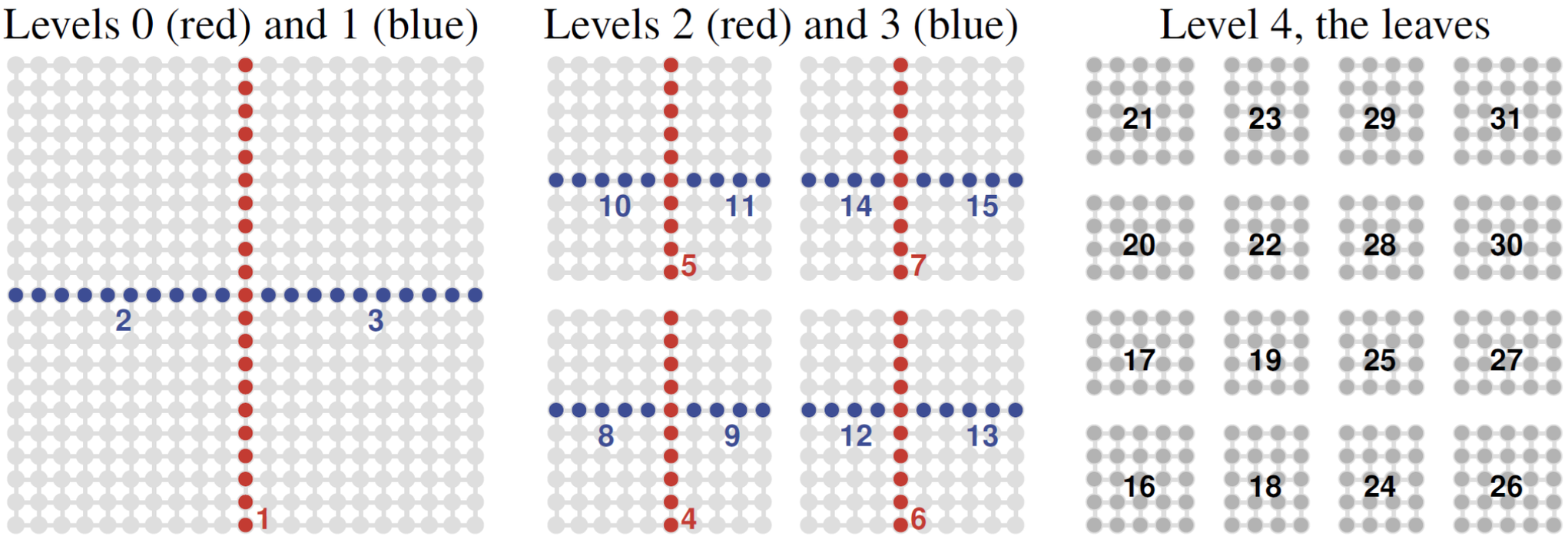
Linear complexity sparse direct solvers: Summary of process

Our model problem remains the Poisson equation discretized on a regular grid:



Linear complexity sparse direct solvers: Summary of process

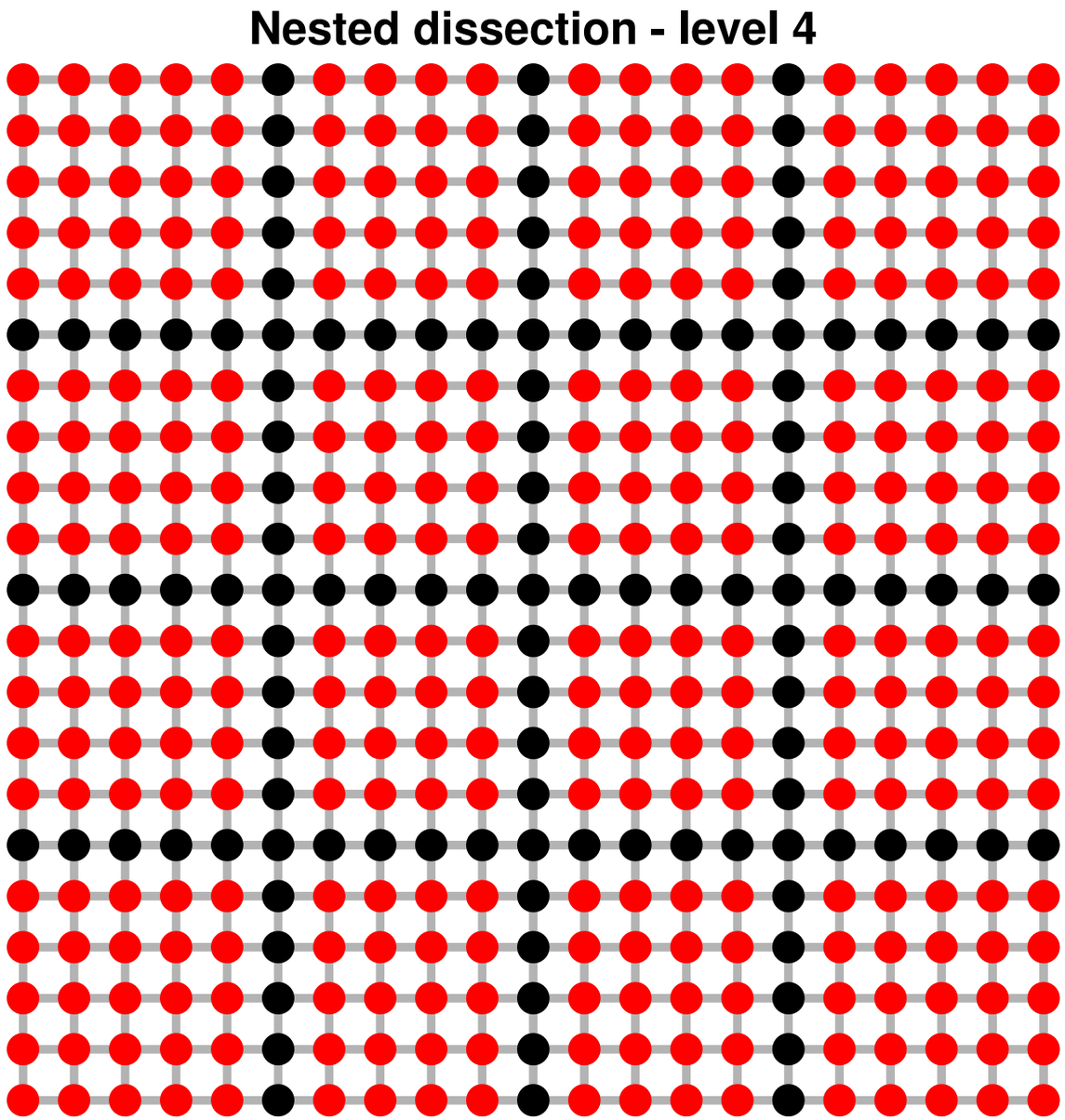
The first thing to do is to create a hierarchical tree based on a nested dissection ordering:



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

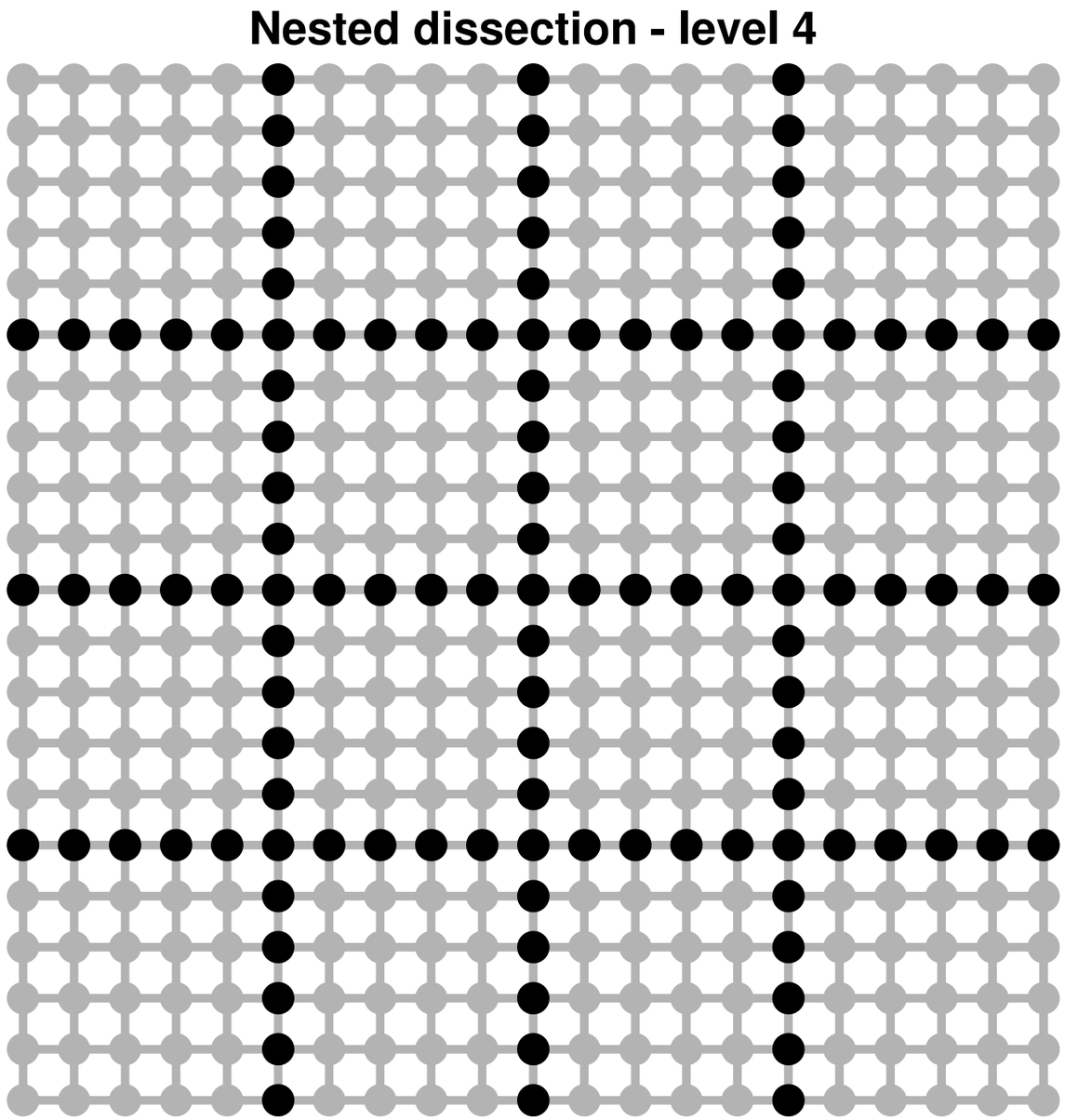
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

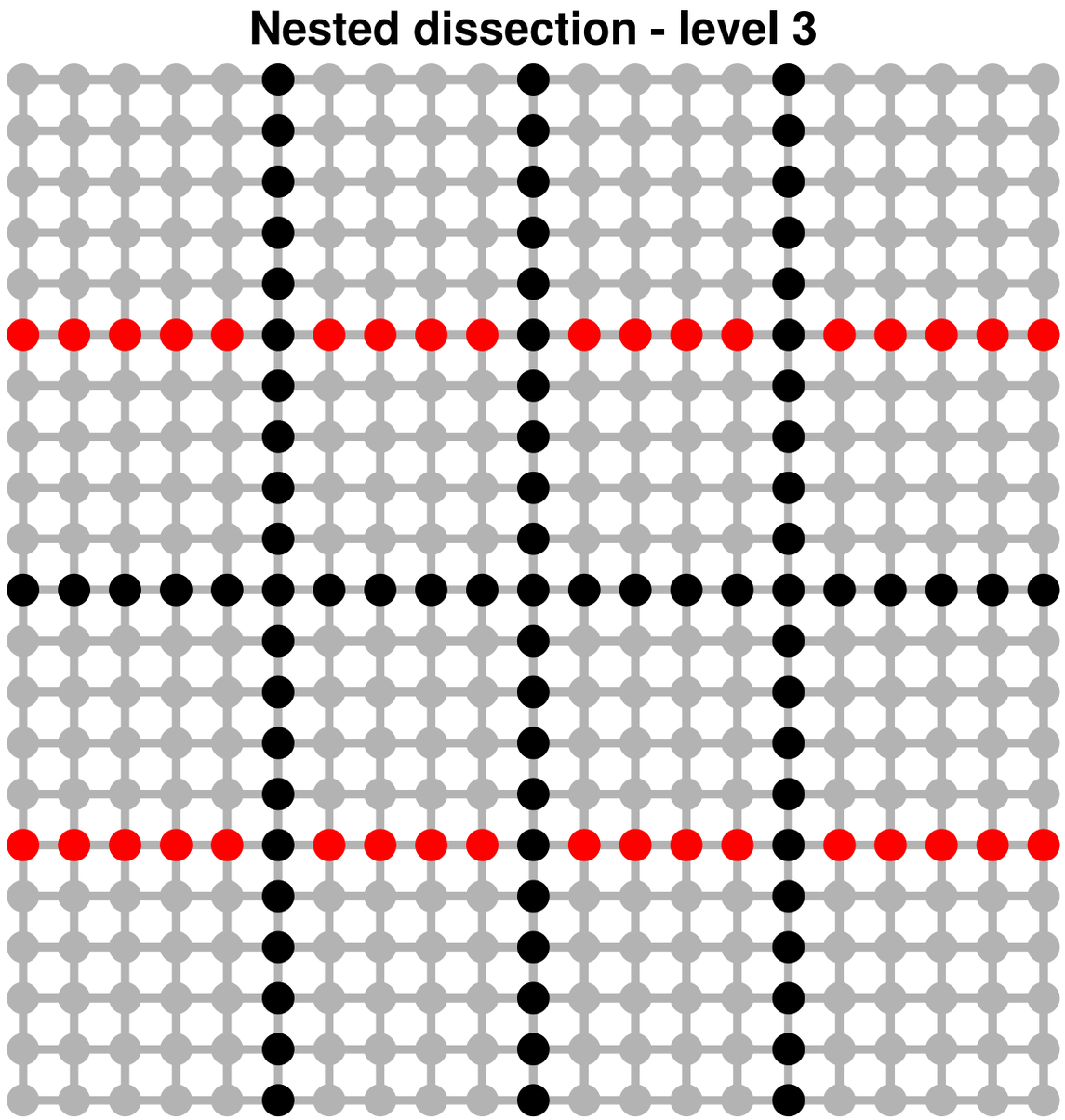
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

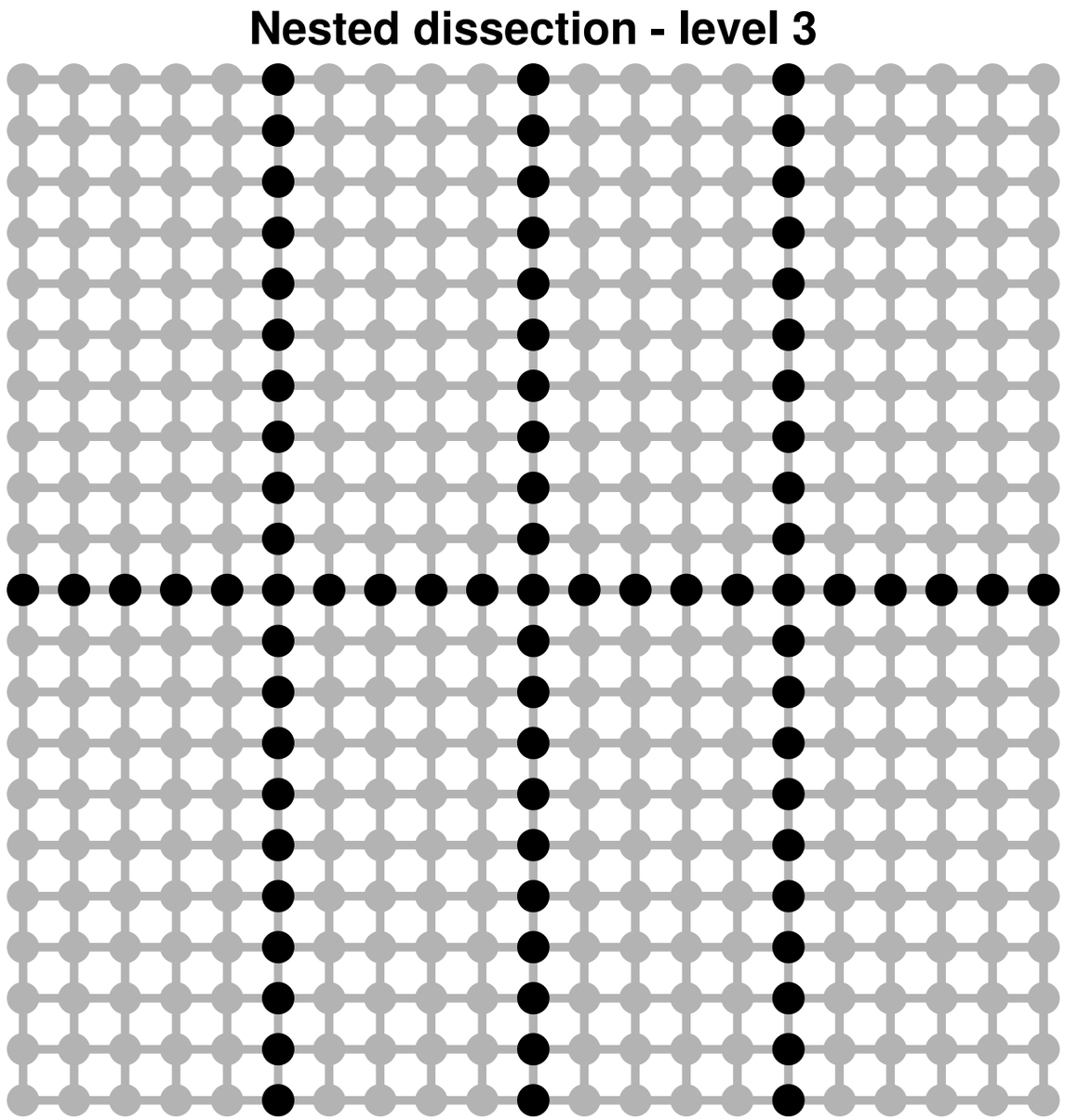
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

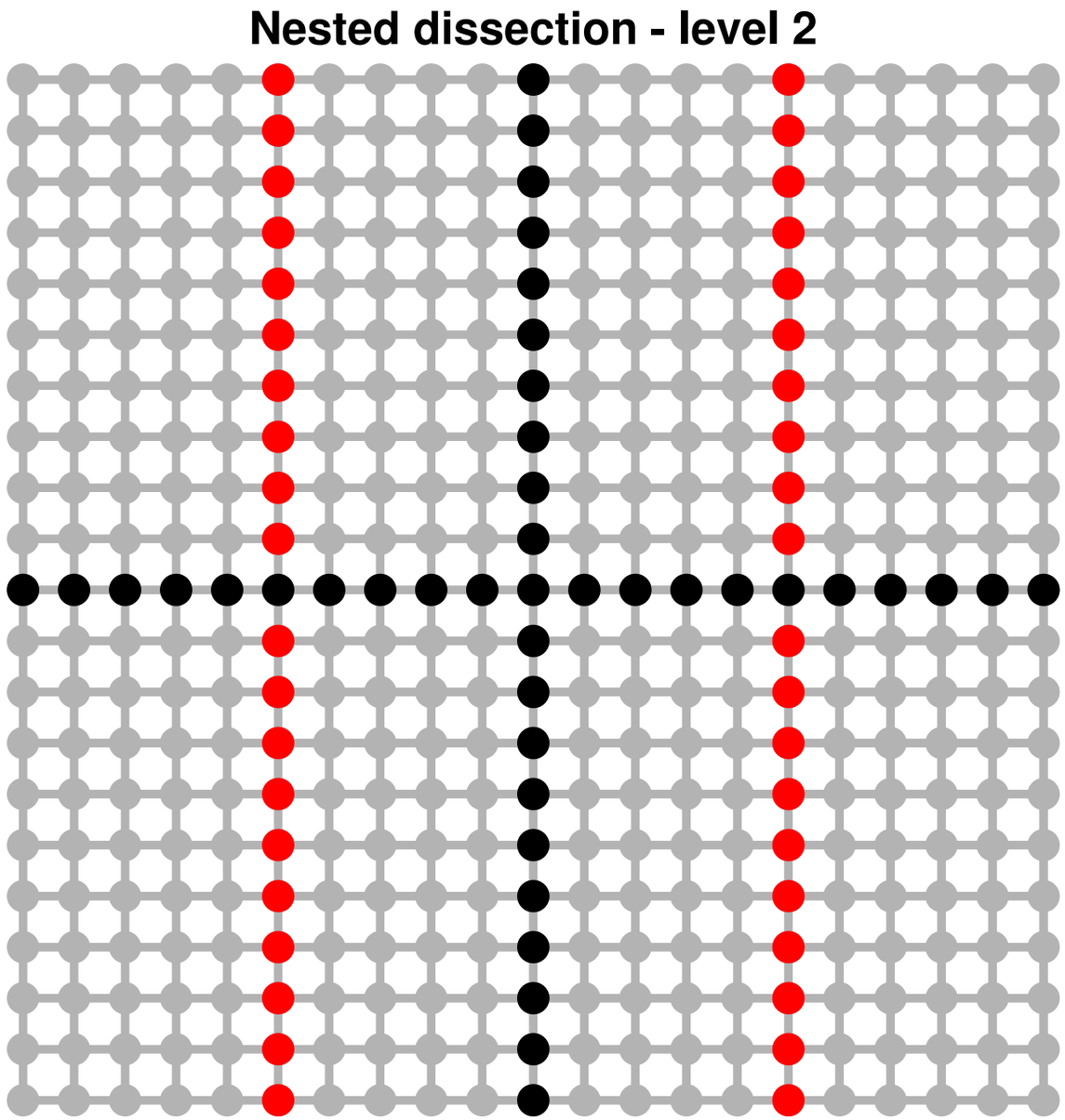
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

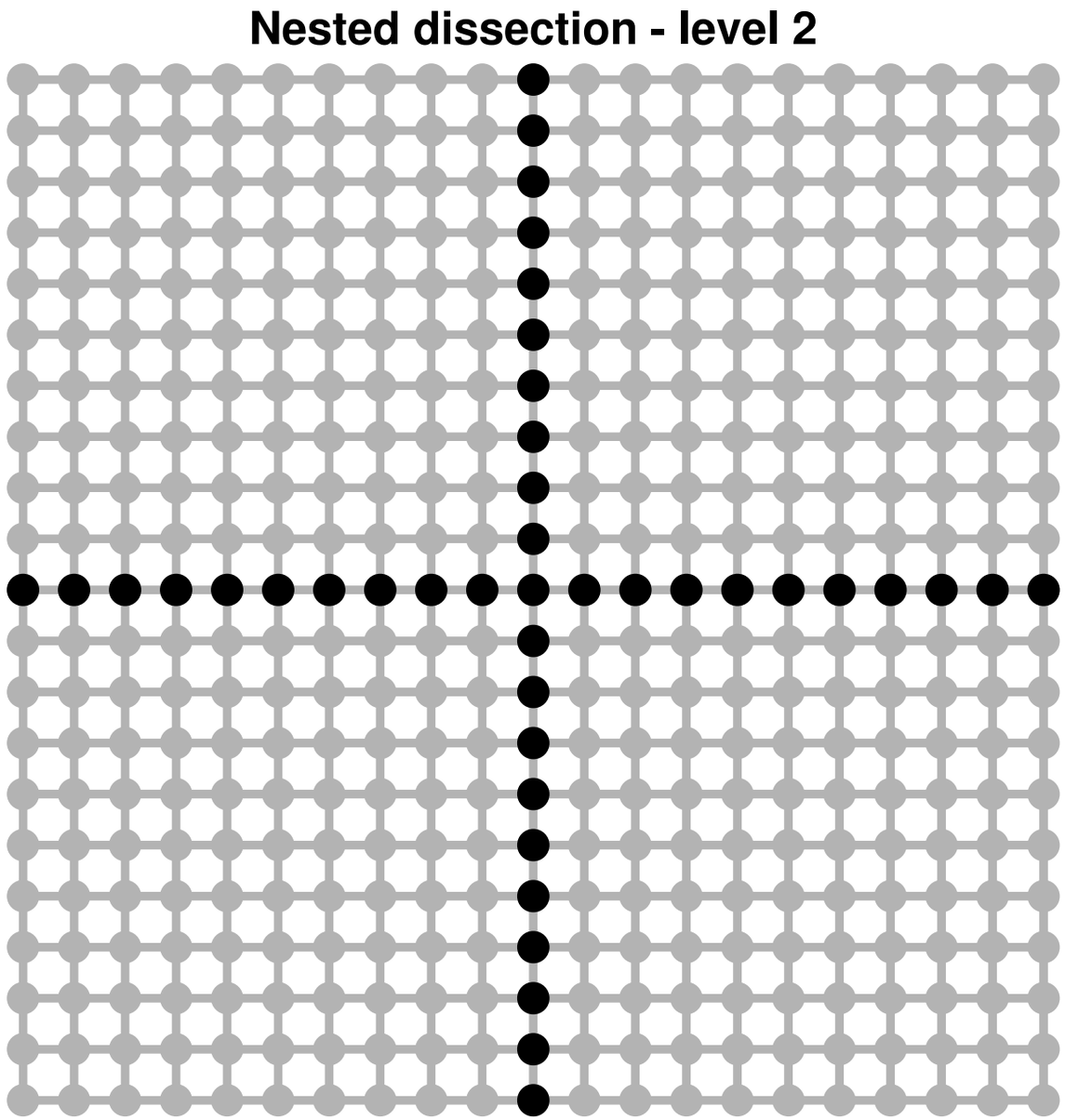
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

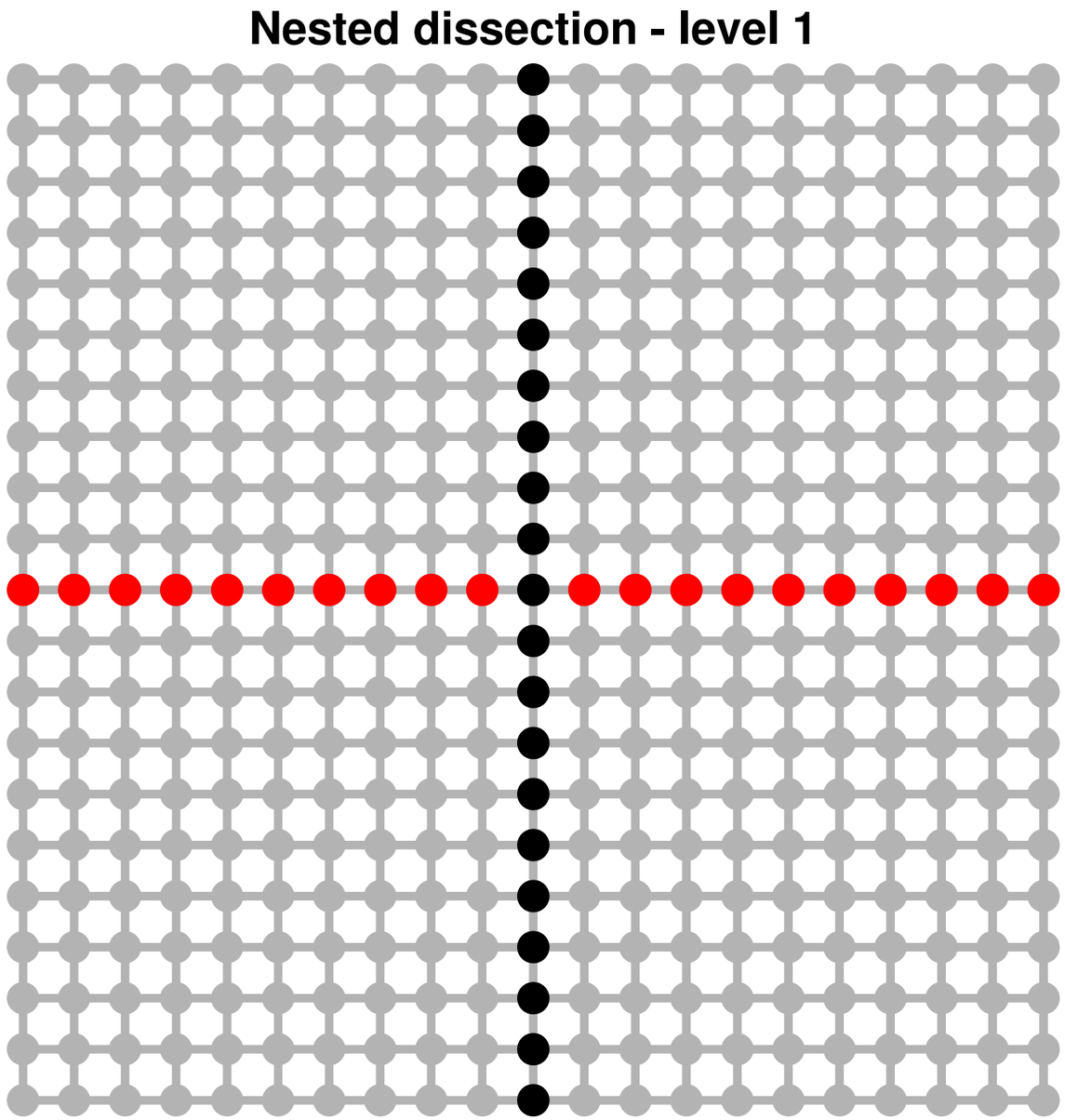
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

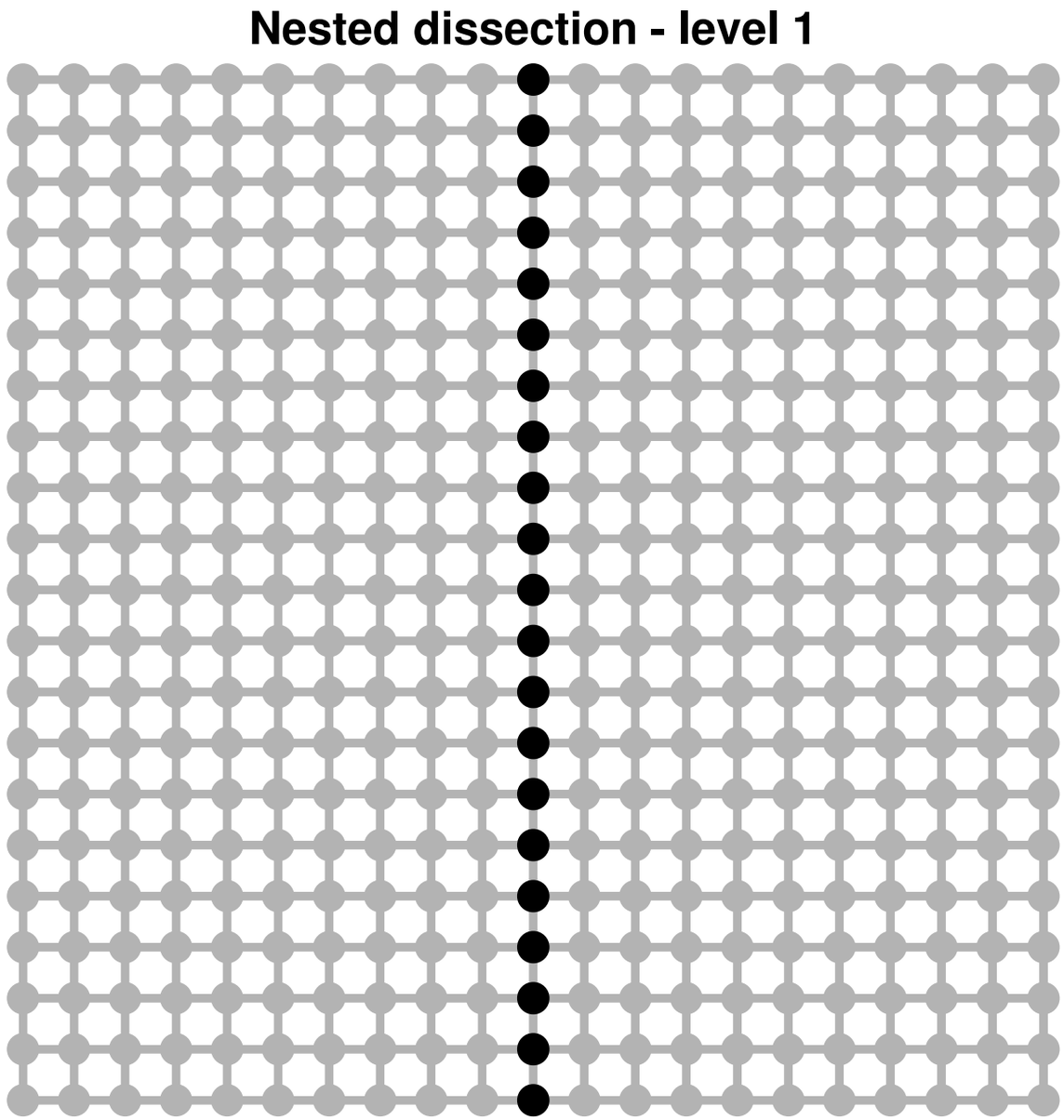
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

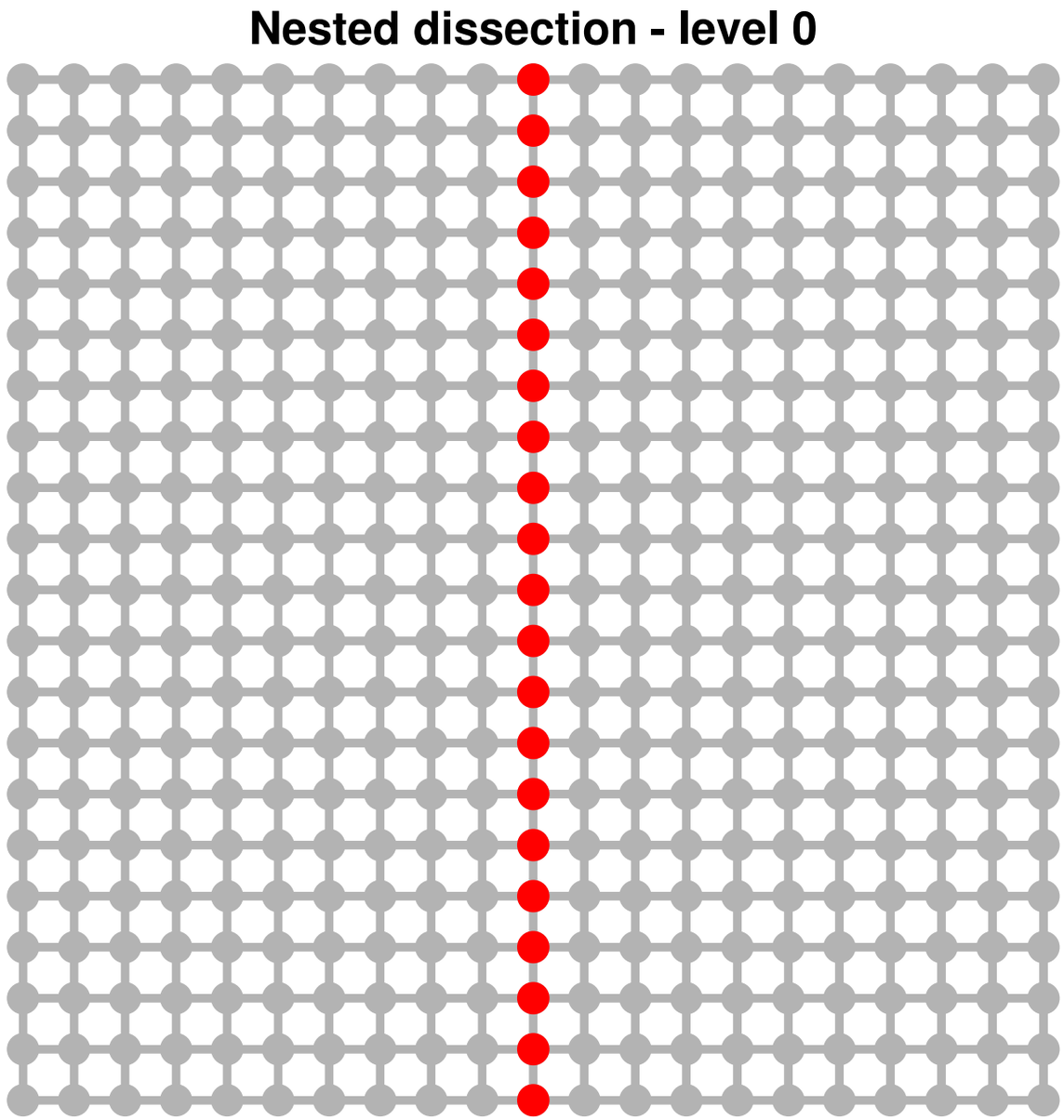
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

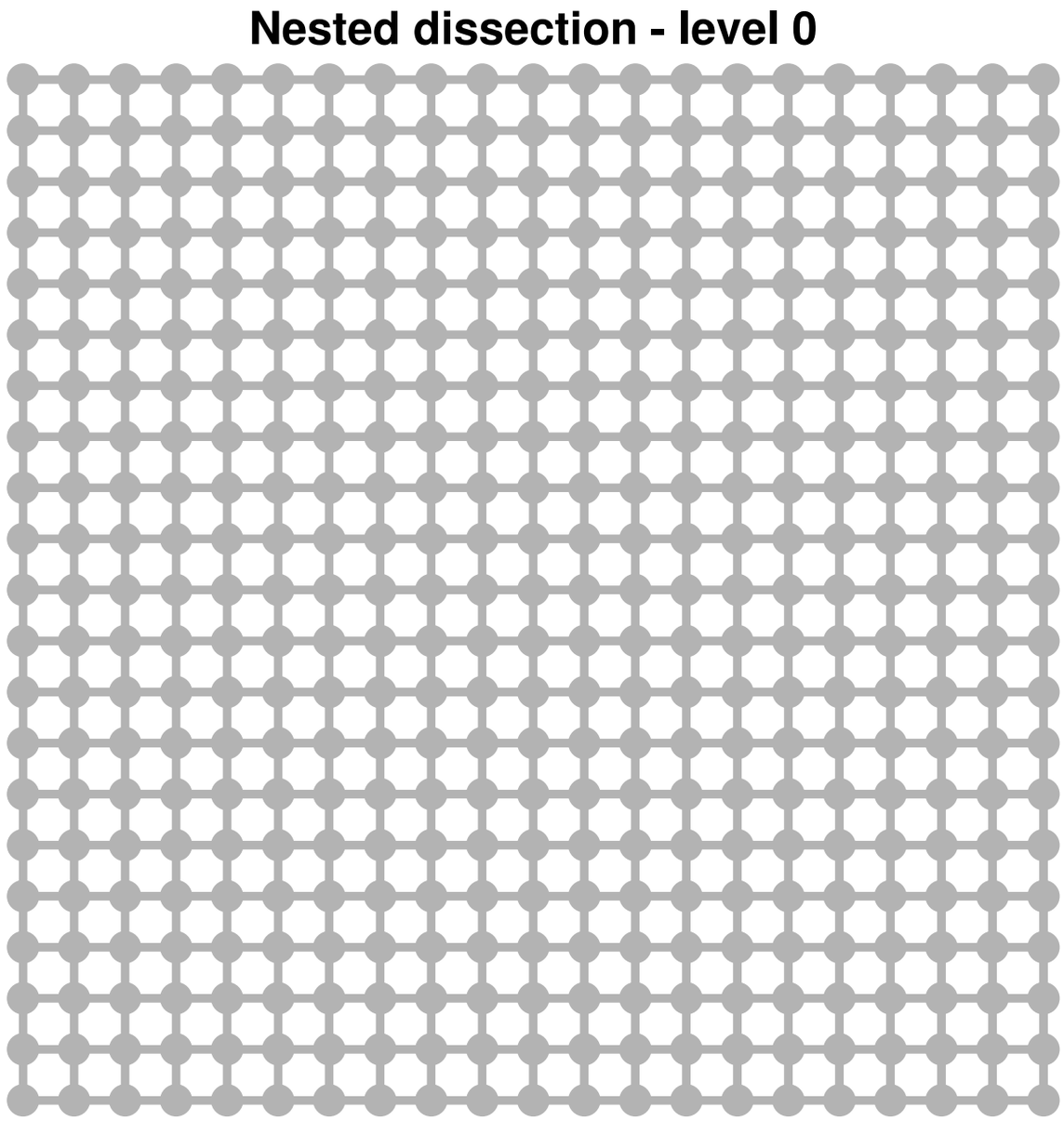
At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers: Summary of process

Sweep through the hierarchical tree, going from smaller to larger boxes.

At each level, eliminate the “active” nodes (red) via Gaussian elimination (local!):



Linear complexity sparse direct solvers

Typically, nested dissection orderings are more complicated:

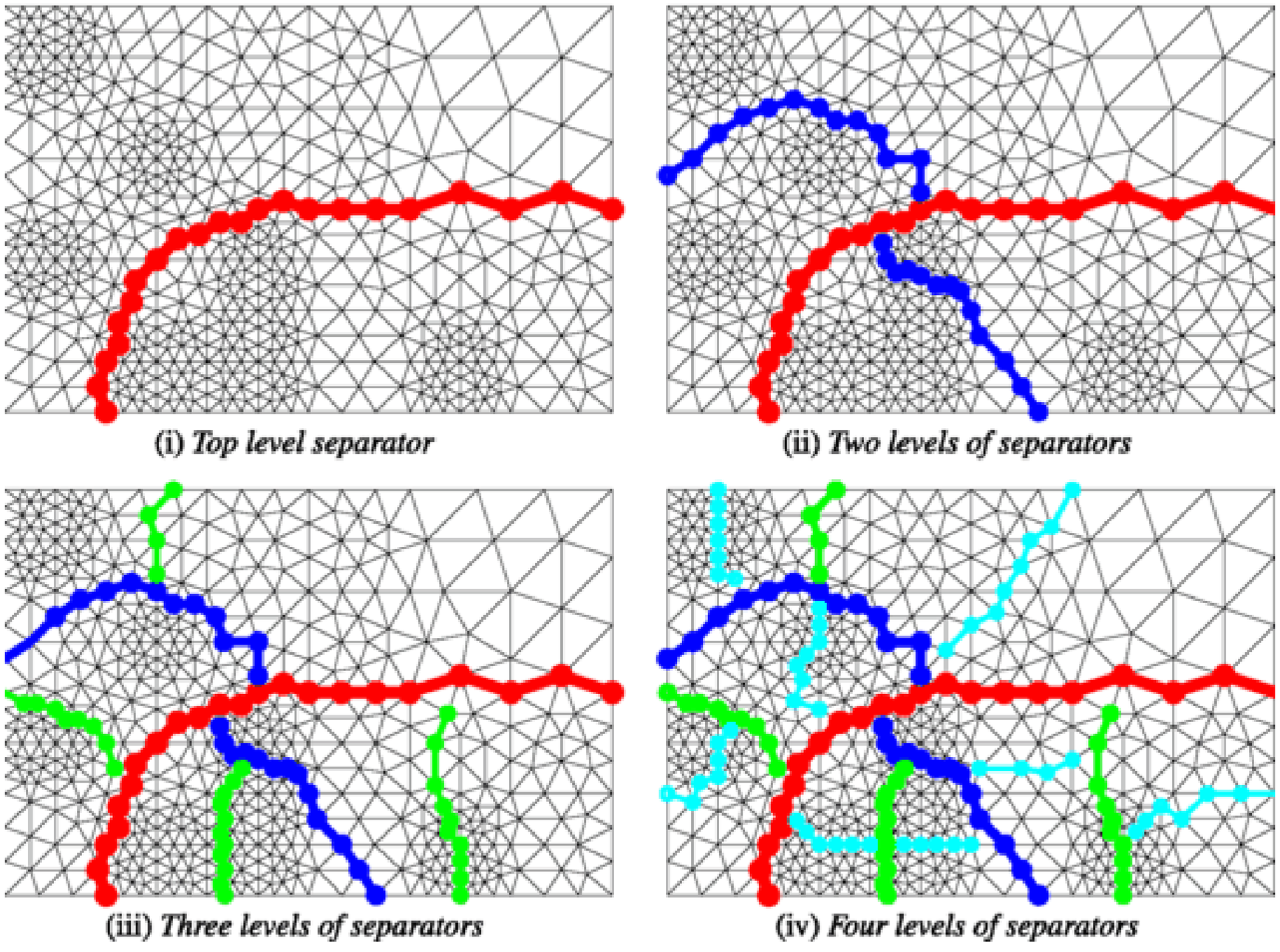


Image credit: Jianlin Xia, "Robust and Efficient Multifrontal Solver for Large Discretized PDEs", 2012

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

Linear complexity sparse direct solvers

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

Alan George

Iain Duff

Tim Davis

Linear complexity sparse direct solvers

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

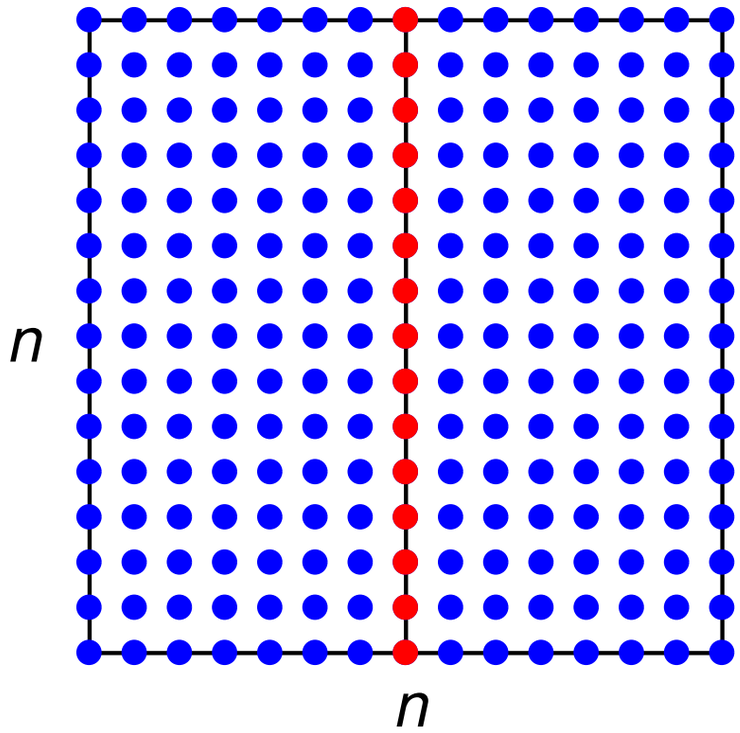
But problems arise as the number of discretization points increases . . .

Linear complexity sparse direct solvers

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

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

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



$$N = n \times n$$

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

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

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

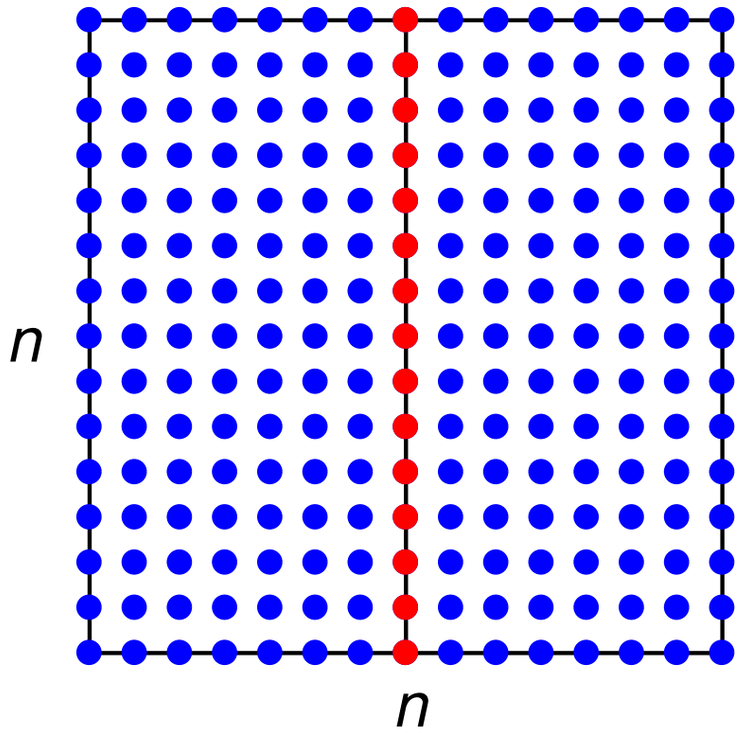
Good news: The $N^{3/2}$ term does not "kick in" until N is huge! Say $N \sim 10^8$.

Linear complexity sparse direct solvers

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

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

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



$$N = n \times n$$

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

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

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

Good news: The $N^{3/2}$ term does not "kick in" until N is huge! Say $N \sim 10^8$.

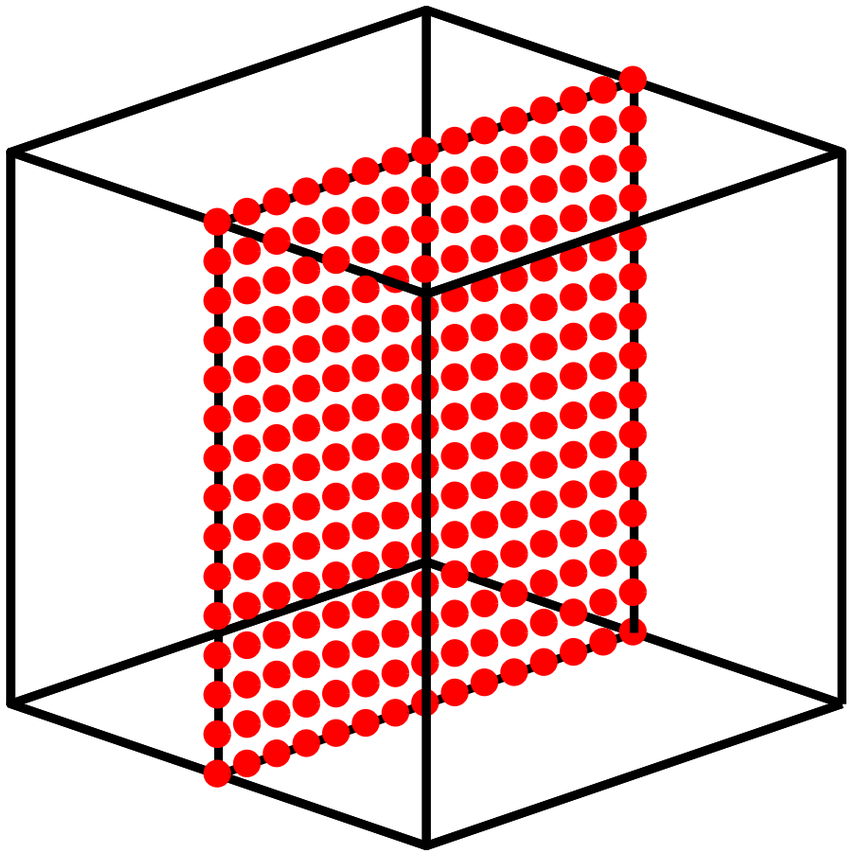
Bad news: *3D is much worse!*

Linear complexity sparse direct solvers

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

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

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



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

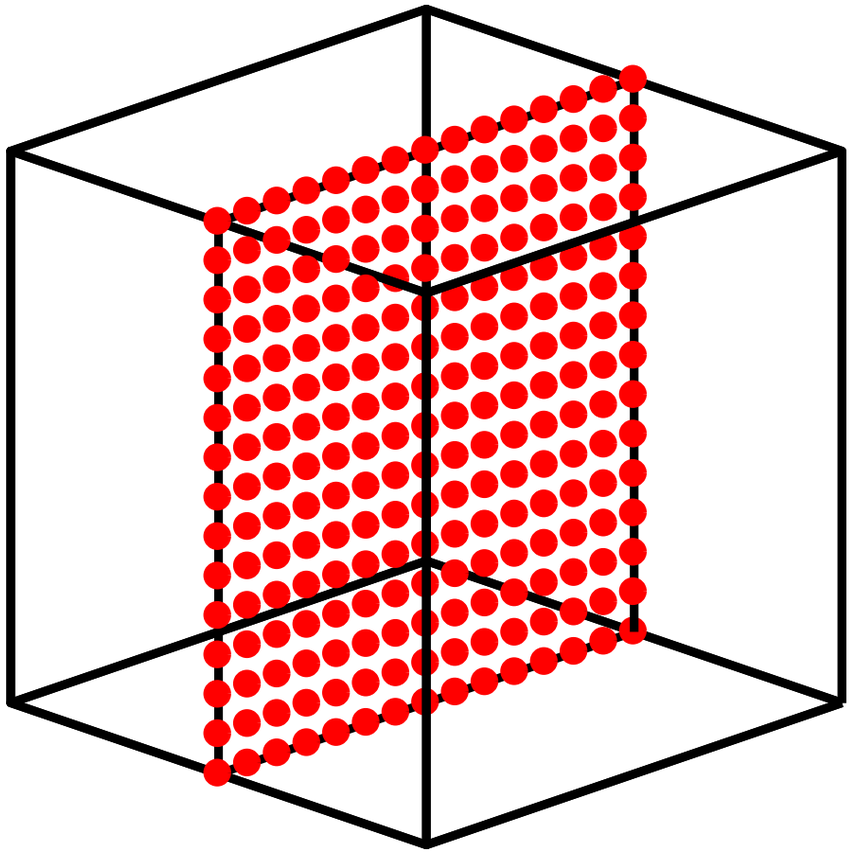
$$\text{COST} \sim (N^{1/3})^6 \sim N^2.$$

Linear complexity sparse direct solvers

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

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

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



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

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

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

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

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

Linear complexity sparse direct solvers

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

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

Linear complexity sparse direct solvers

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

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

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

Linear complexity sparse direct solvers

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

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

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

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

O(N) direct solvers for integral equations were developed by *Martinsson & Rokhlin (2005)*, *Greengard, Gueyffier, Martinsson, & Rokhlin (2009)*, *Gillman, Young, & Martinsson (2012)*, *Ho & Greengard (2012)*, *Ho & Ying (2015)*. Work in 1990's *Y. Chen, P. Starr, V. Rokhlin, L. Greengard, E. Michielssen.* Related to work on $\mathcal{H}$ and $\mathcal{H}^2$ matrix methods (1998 and forwards) by *Börm, Bebendorf, Hackbusch, Khoromskij, Sauter, etc.*

Linear complexity sparse direct solvers

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

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

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

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

Linear complexity sparse direct solvers

Key selling point: Better parallelism

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

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

Observations:

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

Linear complexity sparse direct solvers

Key selling point: Better parallelism

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

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

Observations:

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

Interaction ranks

- Why are they small?
- How small are they, exactly?

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:

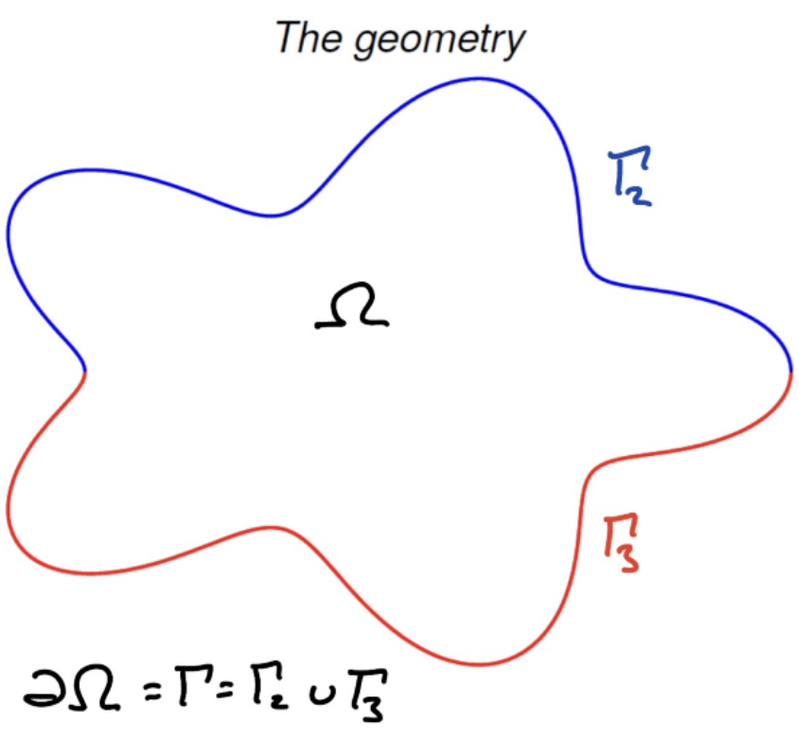
The first is a reformulation of a Dirichlet problem involving the Laplace equation:

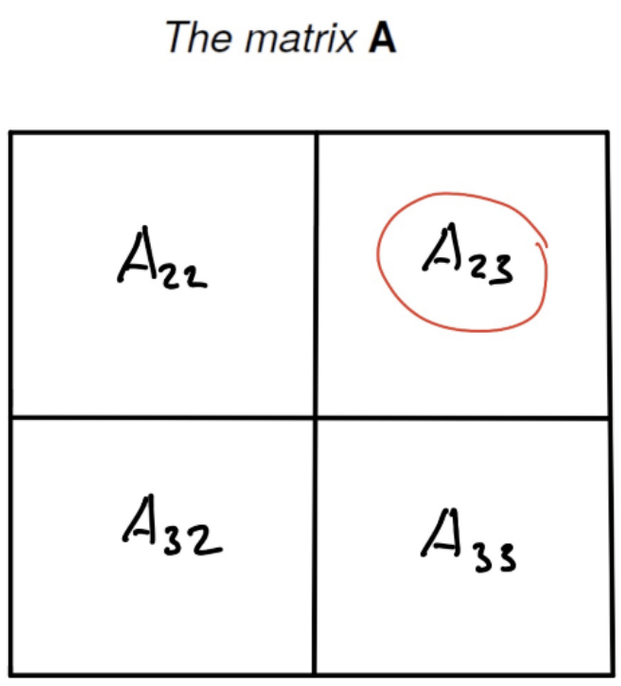
$$\alpha\sigma(\boldsymbol{x}) + \int_\Gamma \big(d(\boldsymbol{x},\boldsymbol{y}) + s(\boldsymbol{x},\boldsymbol{y})\big)\,\sigma(\boldsymbol{y})\,ds(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma.$$

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.

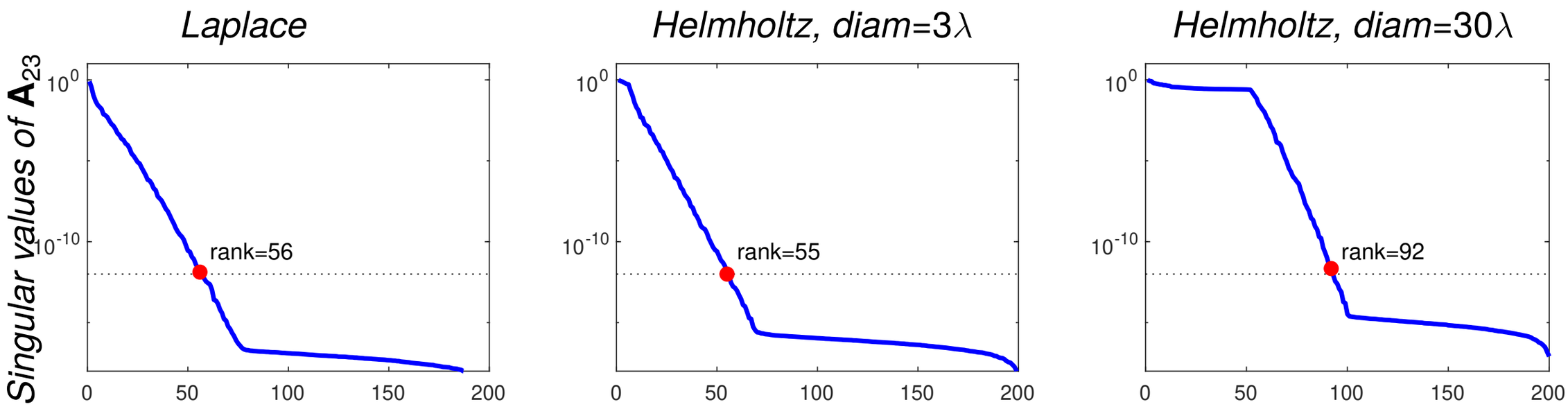




On the next slide, we show the singular values of the off-diagonal block $\mathbf{A}_{23}$.

Interaction ranks: Boundary integral equations

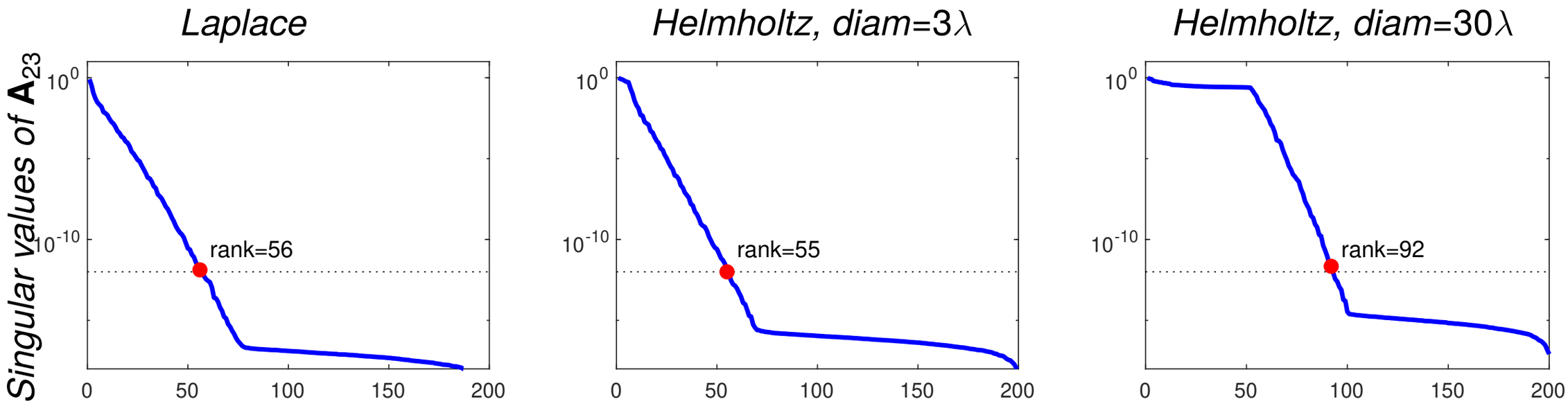
The ranks of an off-diagonal block of **A**:



This is all as expected. Somewhat accessible by analysis.

Interaction ranks: Boundary integral equations

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

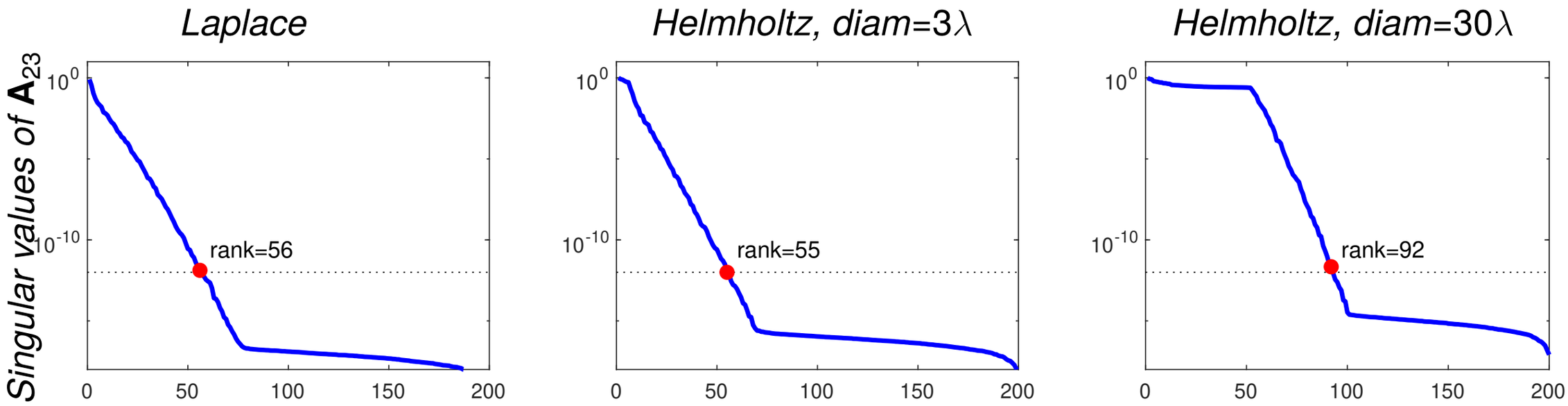


This is all as expected. Somewhat accessible by analysis.

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

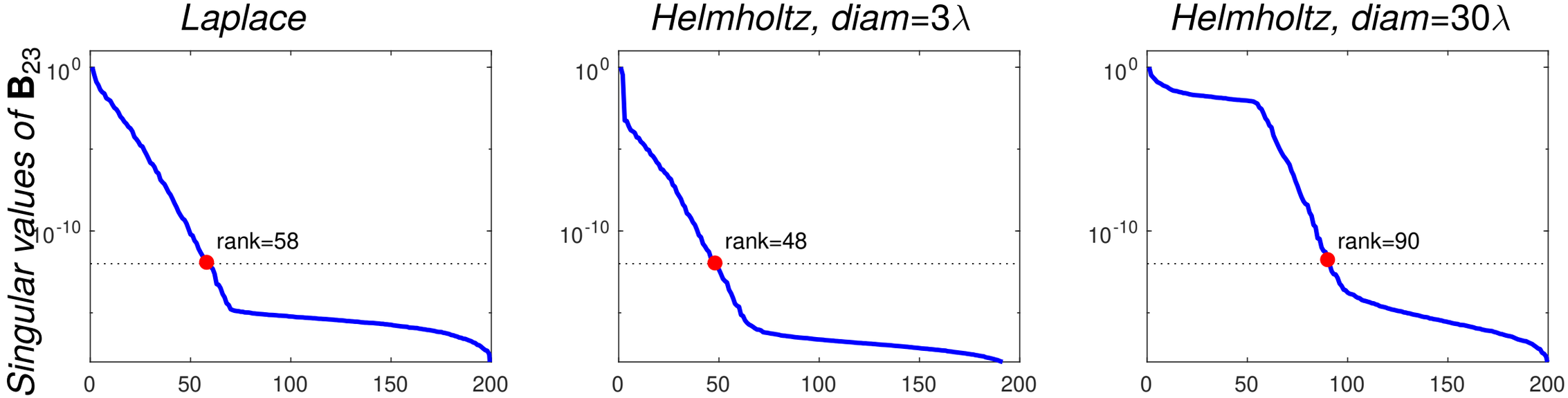
Interaction ranks: Boundary integral equations

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 $\mathbf{B}_{23}$.

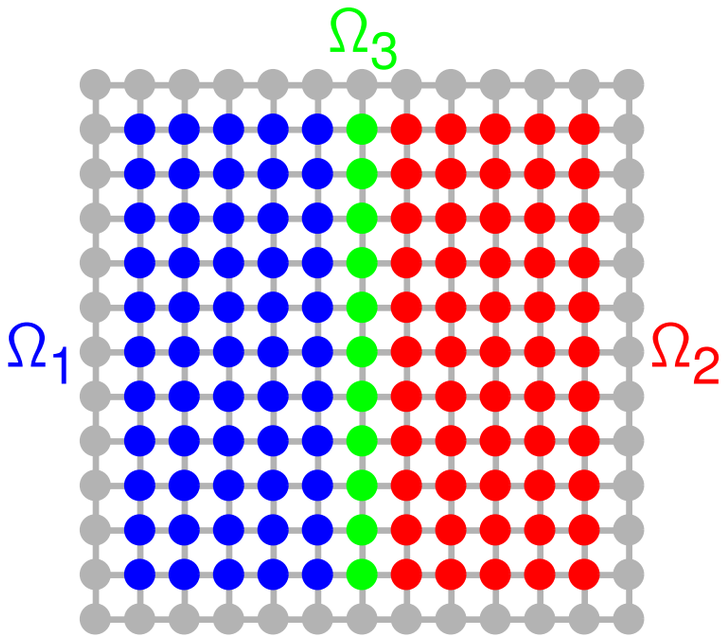


Remarkable similarity!

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

Interaction ranks: Stiffness matrix from finite difference discretization

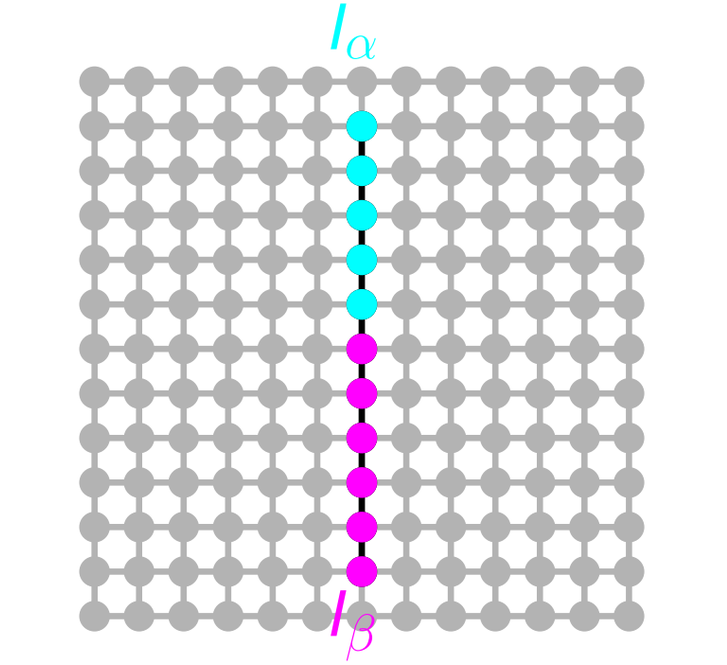
Recall our example of Laplace's equation discretized using the 5-point stencil.



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

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:

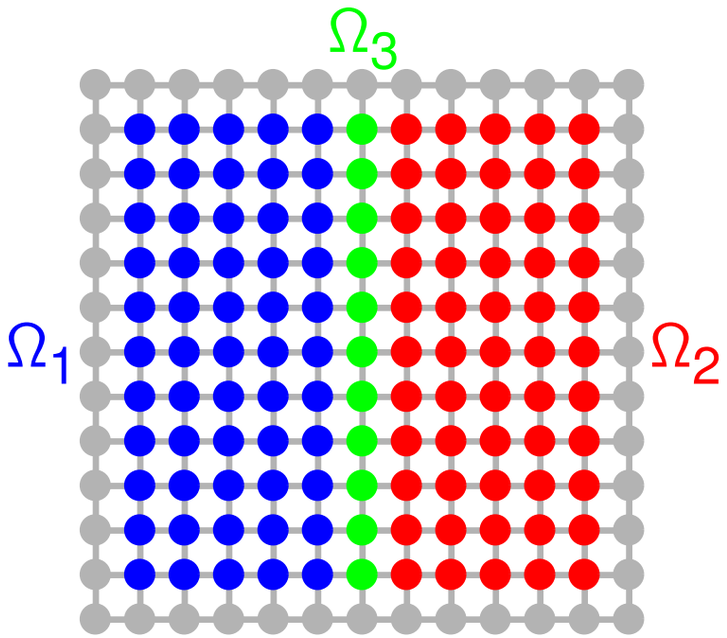


$$\mathbf{S} = \begin{array}{|c|c|} \hline \mathbf{S}_{\alpha\alpha} & \mathbf{S}_{\alpha\beta} \\ \hline \mathbf{S}_{\beta\alpha} & \mathbf{S}_{\beta\beta} \\ \hline \end{array}$$

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

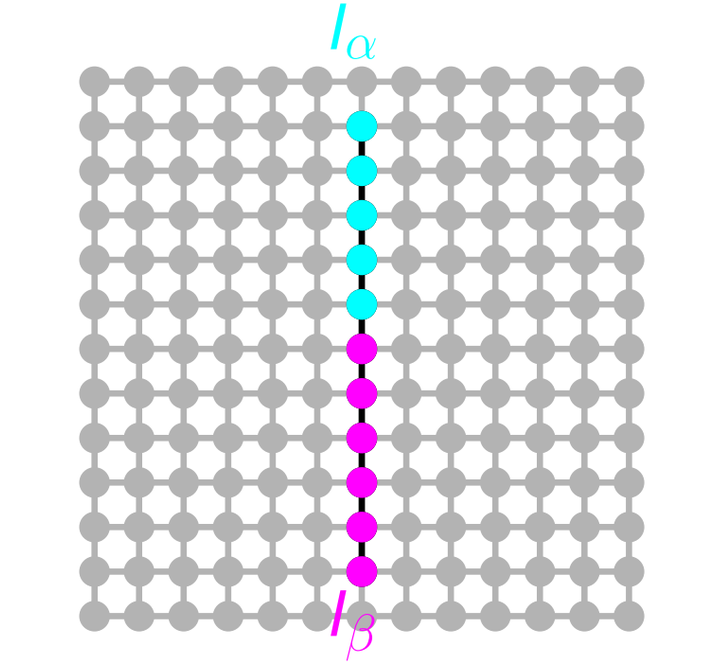
Recall our example of Laplace's equation discretized using the 5-point stencil.



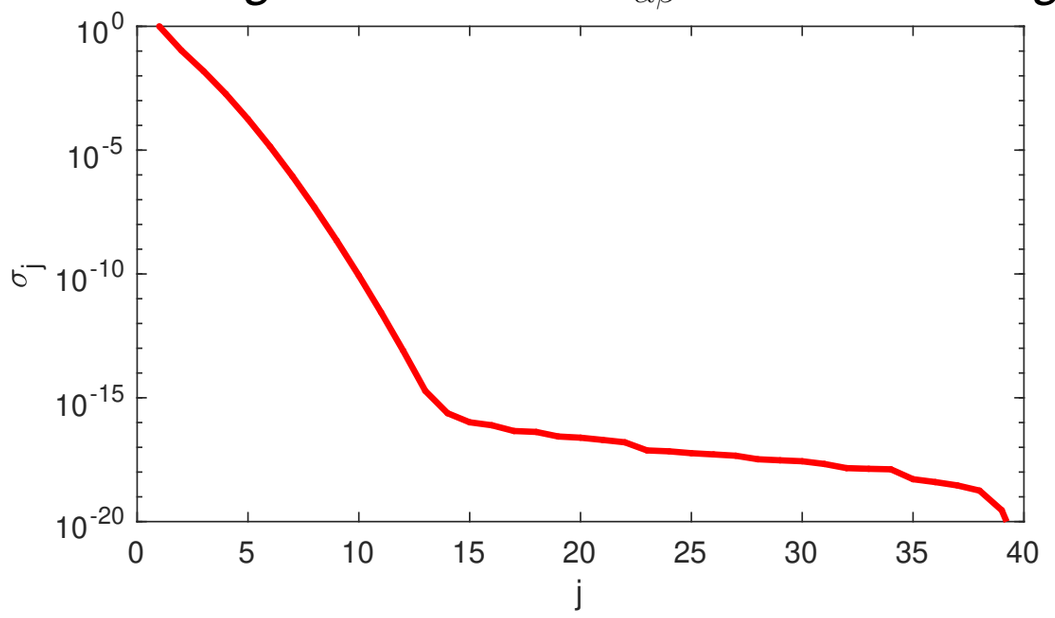
$$\mathbf{A} = \begin{array}{|c|c|c|} \hline \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{13} \\ \hline \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \hline \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \\ \hline \end{array}$$

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:



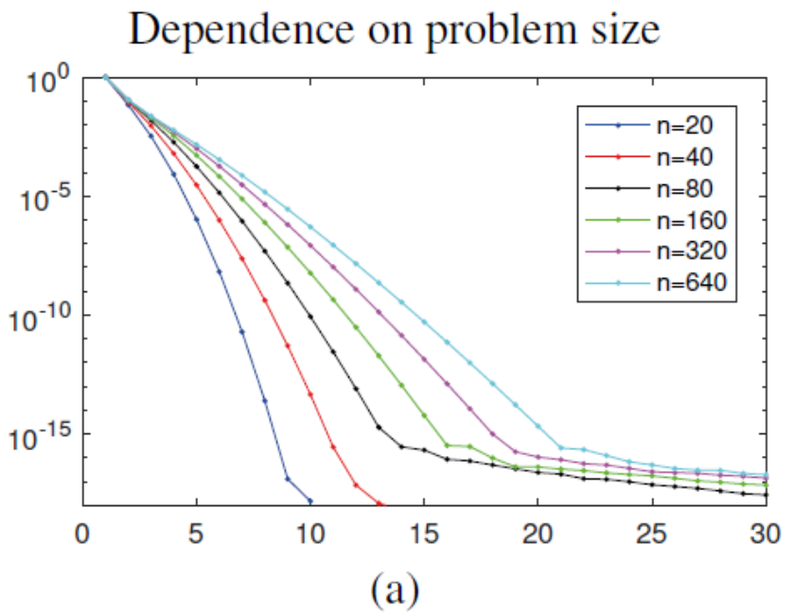
Singular values of $S_{\alpha\beta}$ for an 80×80 *grid.*

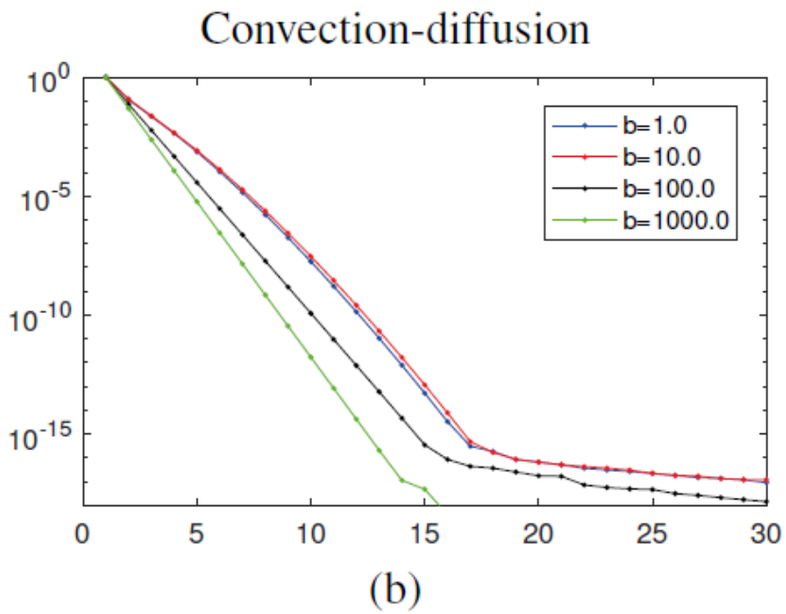


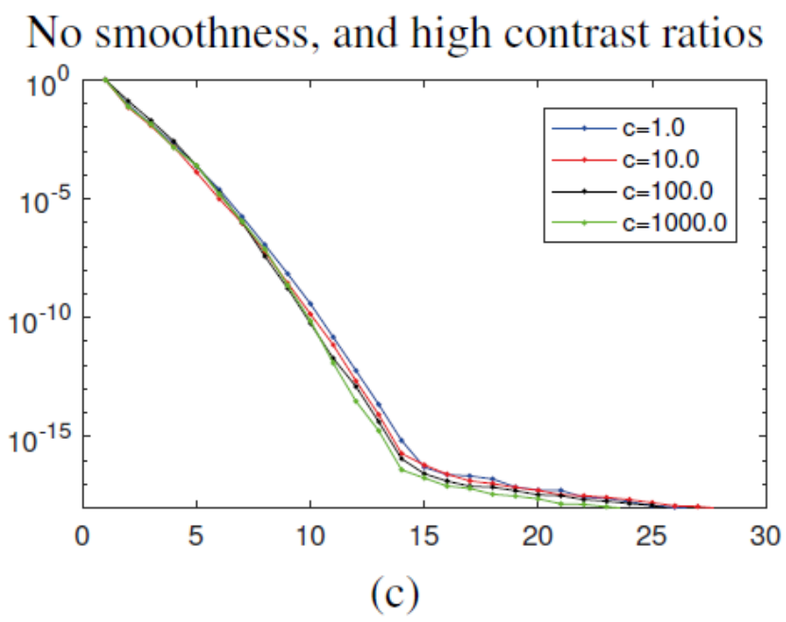
We explore the svds of $\mathbf{S}_{\alpha\beta}$ — encoding interactions between I_α and I_β.

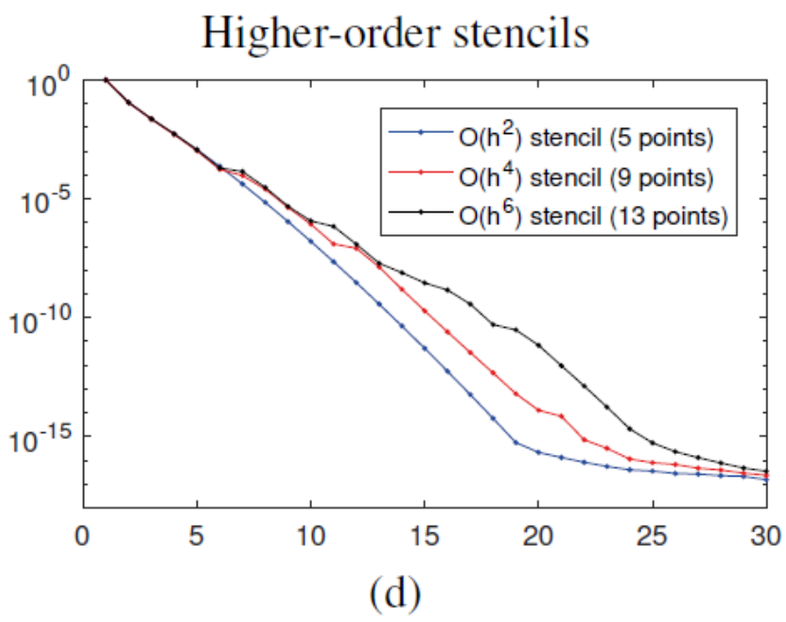
Interaction ranks: Stiffness matrix from finite difference discretization

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









Note: The rank decay property is remarkably stable!

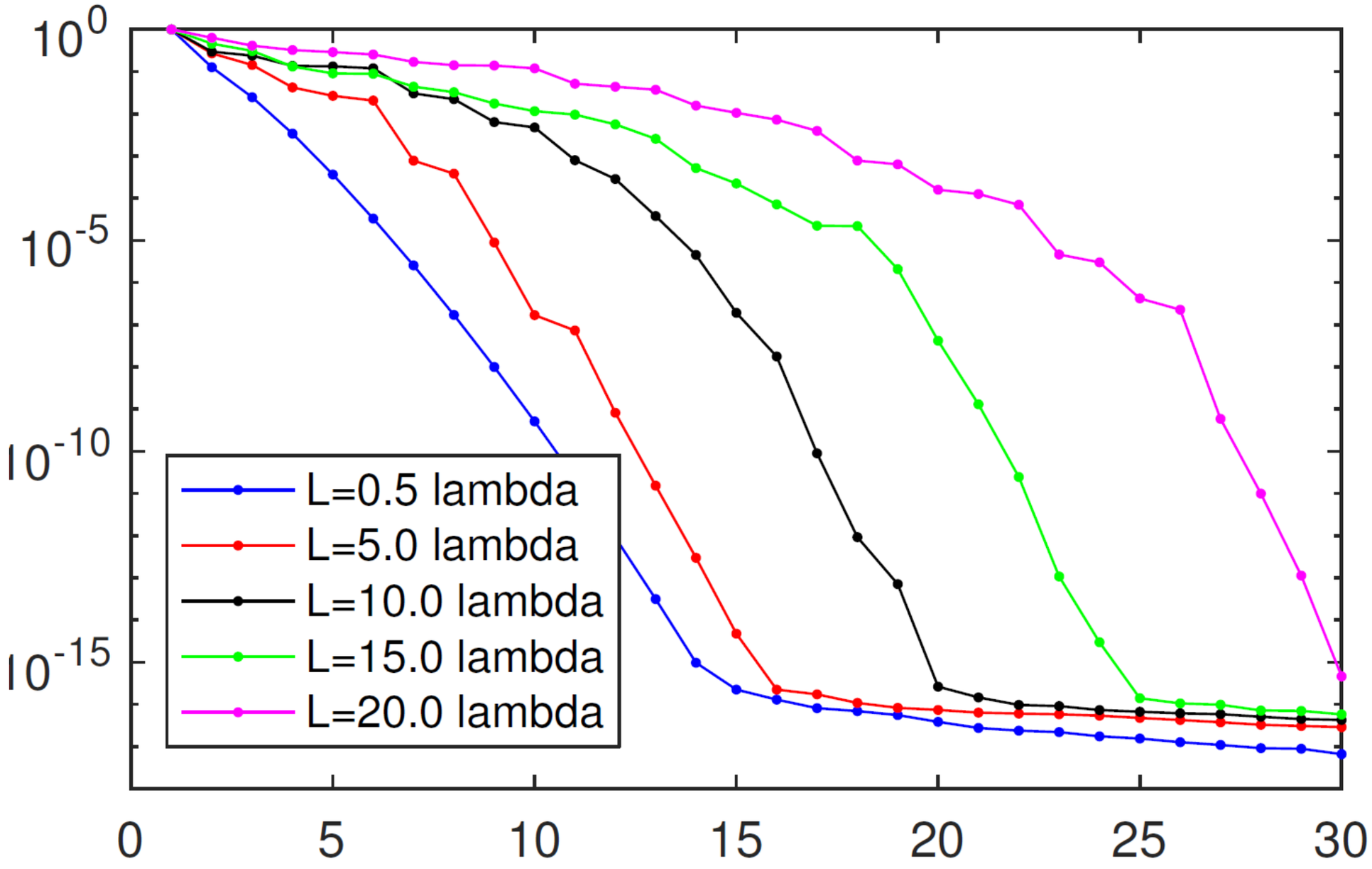
Note: The decay continues to ϵ_{mach} — regardless of the discretization errors!

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

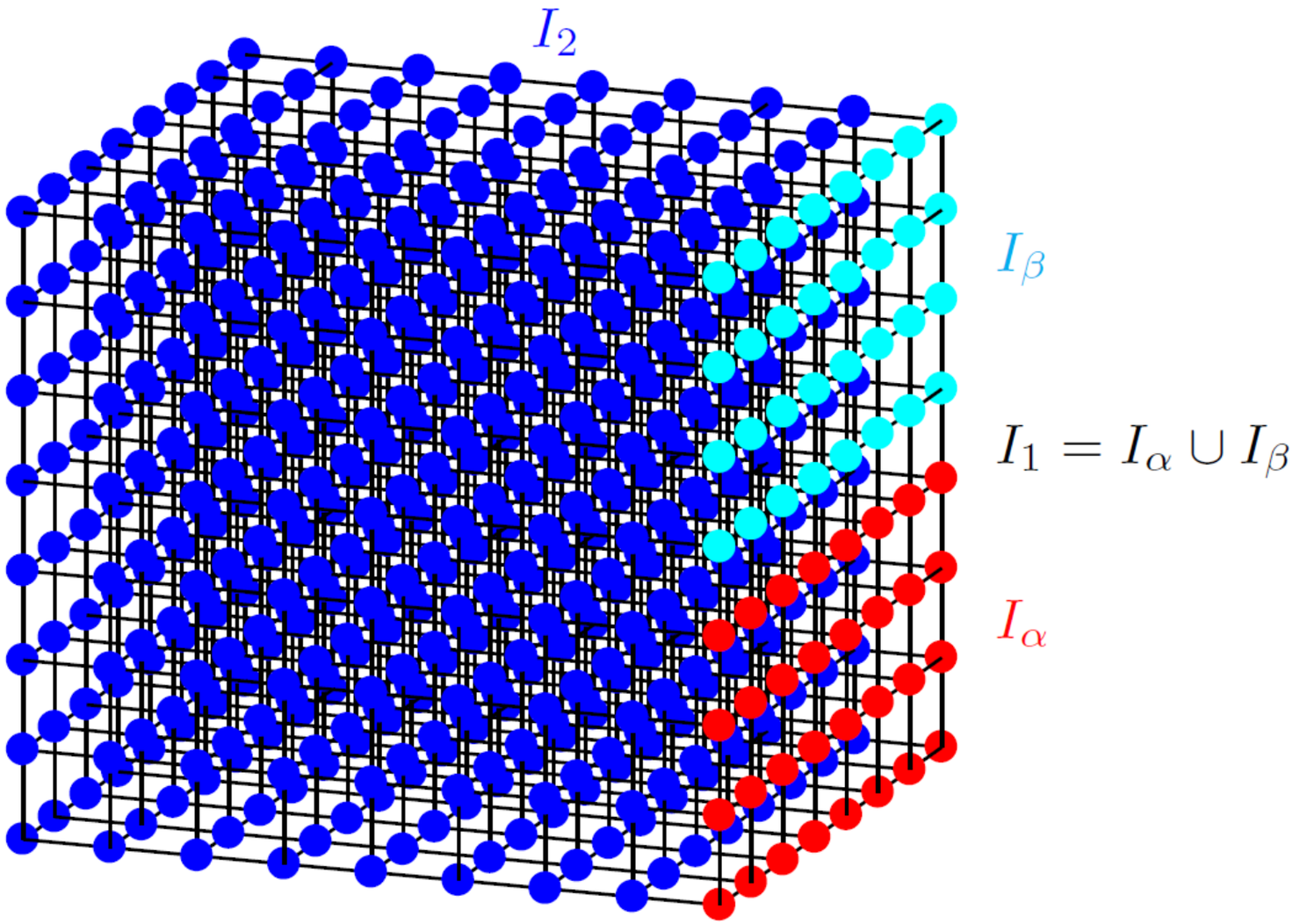
Next, let us consider Helmholtz problems with increasing wave numbers.



Fast decay *once oscillations are resolved.*

Interaction ranks: Stiffness matrix from finite difference discretization

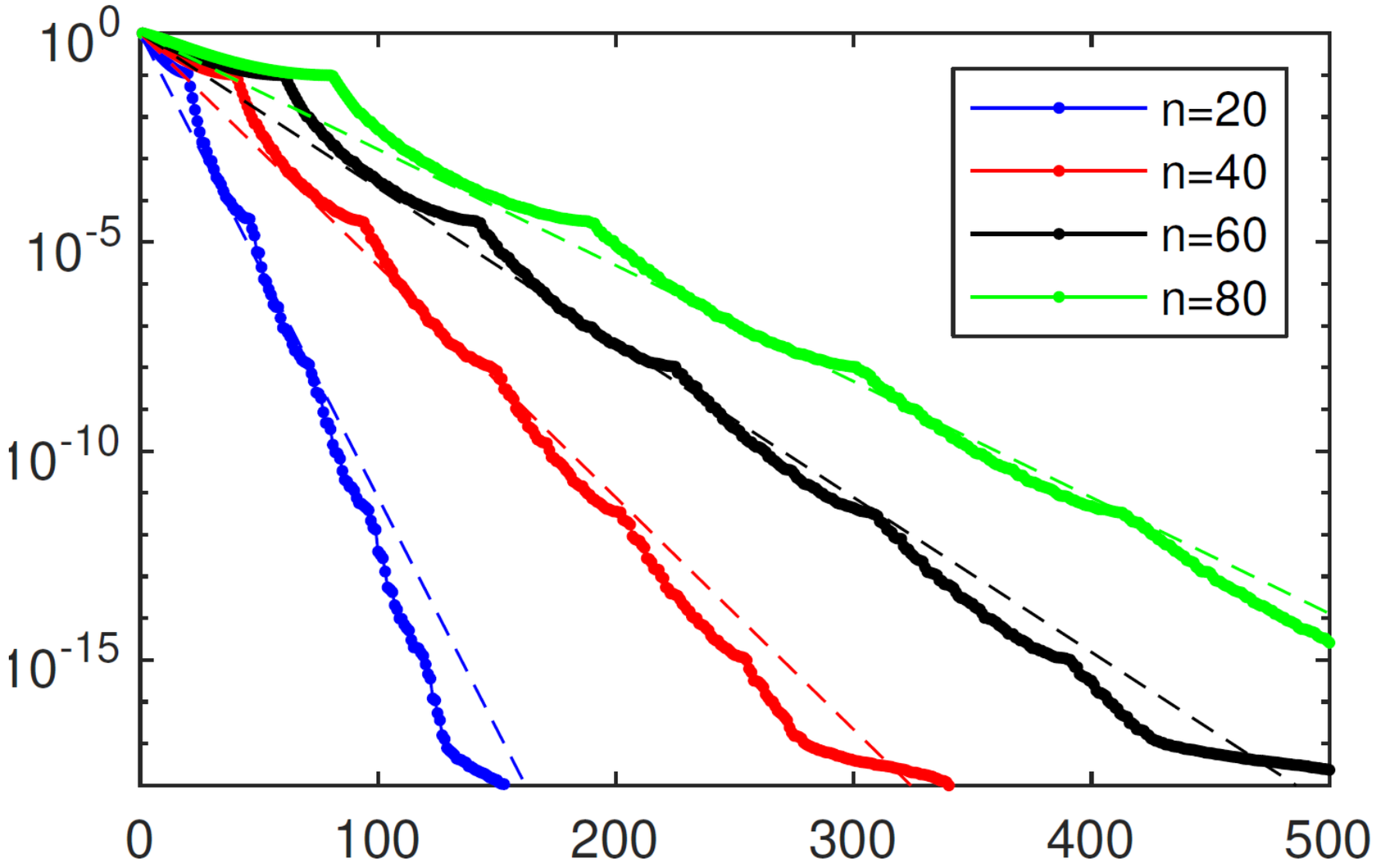
Finally, let us consider the analogous 3D problem.



The geometry.

Interaction ranks: Stiffness matrix from finite difference discretization

Finally, let us consider the analogous 3D problem.



The singular values.