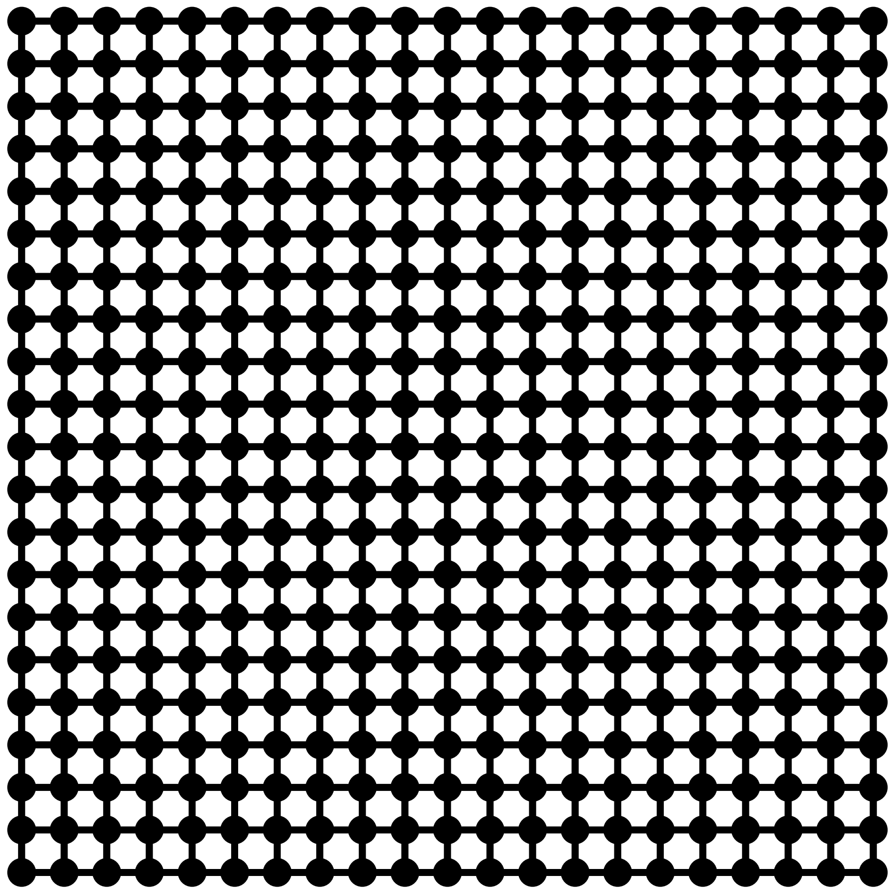
MATH 393C: Fast Methods in Scientific Computing

Lecture on April 16, 2019

P.G. Martinsson

The University of Texas at Austin

A model problem



Let $\Omega = [0,1]^2$. We introduce a square $n \times n$ grid on Ω with nodes $\{\boldsymbol{x}_j\}_{j=1}^N$ where $N = n^2$.

Let $\mathbf{u} = [\mathbf{u}(j)]_{j=1}^N$ denote a potential vector, and let $\mathbf{f} = [\mathbf{f}(j)]_{j=1}^N$ denote a given load vector.

The equilibrium equation we seek to solve is

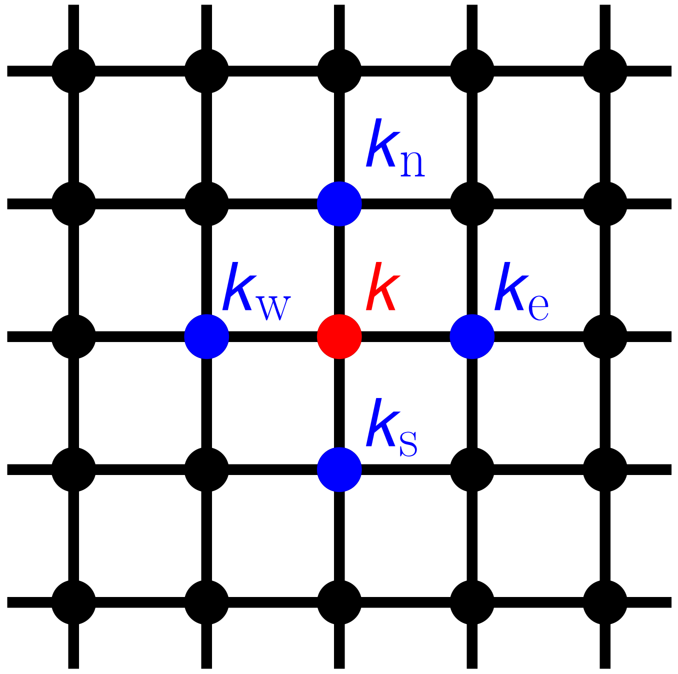
$$\mathbf{Au} = \mathbf{f},$$

where $\mathbf{A}$ is a finite difference operator discretizating an elliptic boundary value problem on Ω (Laplace, Helmholtz, convection-diffusion, etc).

The archetypical example of an elliptic finite difference equation is the 5-point stencil

$$[\mathbf{Au}](k) = \frac{1}{h^2}\left(4\,\mathbf{u}(k) - \mathbf{u}(k_{\mathrm{s}}) - \mathbf{u}(k_{\mathrm{e}}) - \mathbf{u}(k_{\mathrm{n}}) - \mathbf{u}(k_{\mathrm{w}})\right), \tag{1}$$

where h is the grid spacing ($h = 1/(n-1)$), where k is a node in the mesh, and where $\{k_{\mathrm{s}},\, k_{\mathrm{e}},\, k_{\mathrm{n}},\, k_{\mathrm{w}}\}$ are the nodes that are immediate neighbors of k to the south, east, north, and west of k, respectively.



Then

$$\mathbf{Au} = \mathbf{f}$$

is a discrete analog of the Poisson equation (with $\Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$)

$$-\Delta u = f.$$

Example: As a generalization, one could consider a grid conduction problem.

$$[\mathbf{Au}](k) = \alpha_{k,k_{\rm e}}\left(\mathbf{u}(k)-\mathbf{u}(k_{\rm e})\right)+\alpha_{k,k_{\rm n}}\left(\mathbf{u}(k)-\mathbf{u}(k_{\rm n})\right)+\alpha_{k,k_{\rm w}}\left(\mathbf{u}(k)-\mathbf{u}(k_{\rm w})\right)+\alpha_{k,k_{\rm s}}\left(\mathbf{u}(k)-\mathbf{u}(k_{\rm s})\right),$$

where $\alpha_{k,\ell}$ is the conductivity of the link connecting nodes k and ℓ.

Example: Another option is to consider a more general elliptic PDE

$$-\Delta u(\boldsymbol{x}) + (b(\boldsymbol{x}), c(\boldsymbol{x}))\cdot\nabla u(\boldsymbol{x}) + d(\boldsymbol{x})u(\boldsymbol{x}) = f(\boldsymbol{x}).$$

In this case

$$\begin{aligned}[\mathbf{Au}](k) =& \frac{1}{h^2}\left(4\mathbf{u}(k) - \mathbf{u}(k_{\rm e}) - \mathbf{u}(k_{\rm n}) - \mathbf{u}(k_{\rm w}) - \mathbf{u}(k_{\rm s})\right)+\\ & b(\mathbf{x}_k)\,\frac{1}{h}\big(\mathbf{u}(k_{\rm e}) - \mathbf{u}(k_{\rm w})\big) + c(\mathbf{x}_k)\,\frac{1}{h}\big(\mathbf{u}(k_{\rm n}) - \mathbf{u}(k_{\rm s})\big) + d(\mathbf{x}_k)\,\mathbf{u}(k).\end{aligned}$$

Spectral properties of A

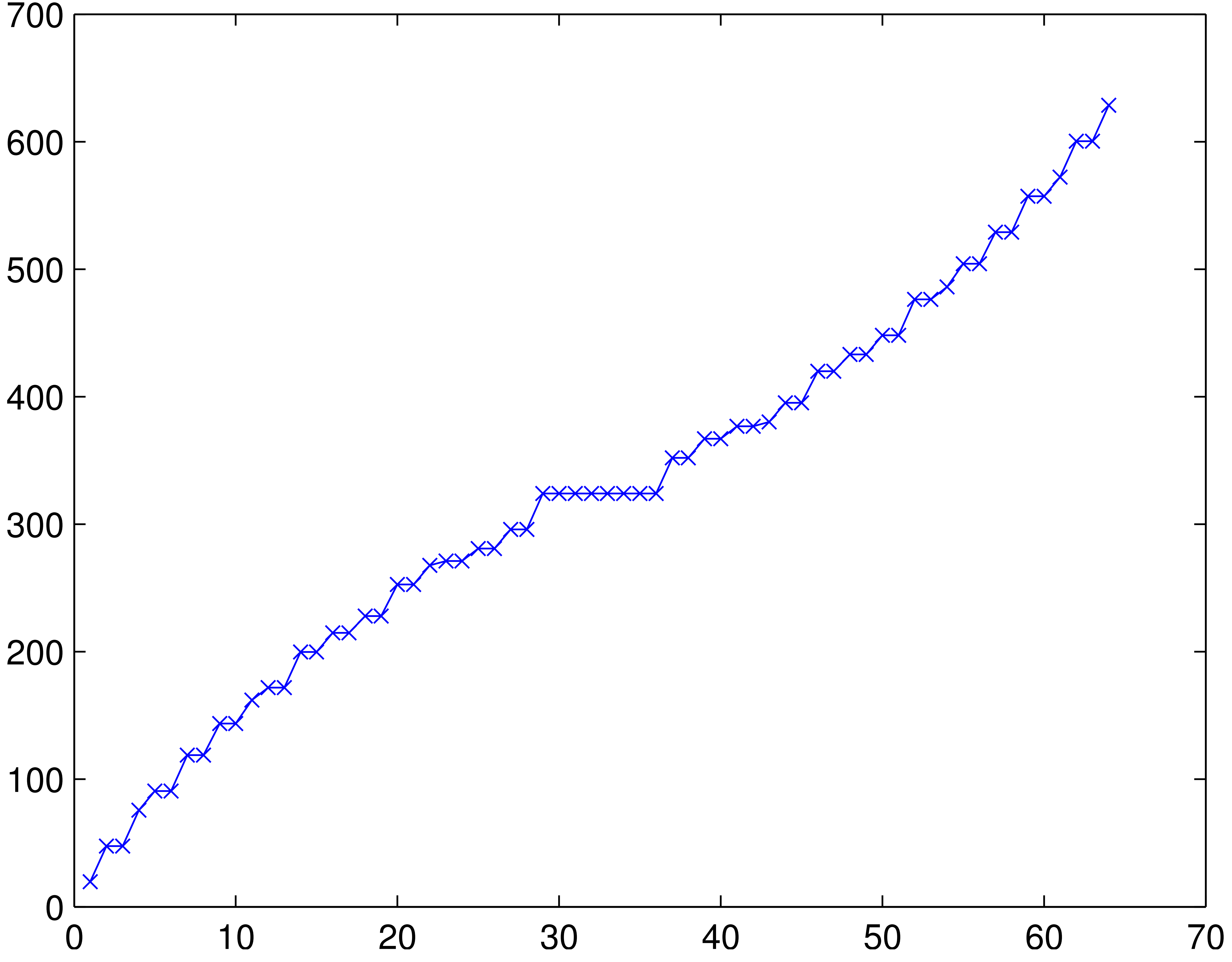
Let **A** denote the standard five-point stencil:

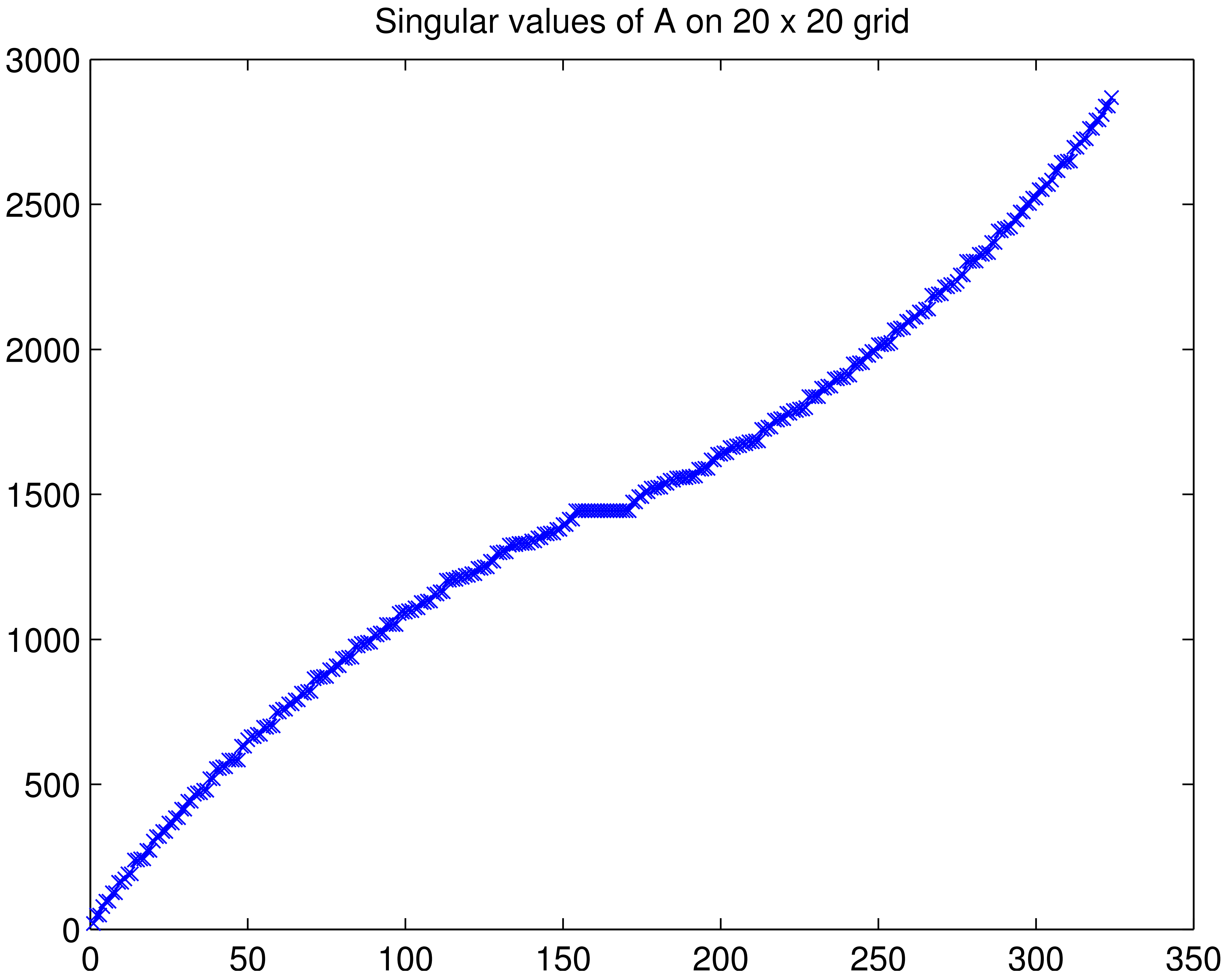
$$[\mathbf{A}\mathbf{u}](k) = \frac{1}{h^2}\big(4\mathbf{u}(k) - \mathbf{u}(k_{\rm e}) - \mathbf{u}(k_{\rm n}) - \mathbf{u}(k_{\rm w}) - \mathbf{u}(k_{\rm s})\big),$$

and let $\sigma_j = \sigma_j(\mathbf{A})$ denote the j'th singular value of **A**.
(We temporarily order them "backwards" so that $0 \leq \sigma_1 \leq \sigma_2 \leq \sigma_3 \leq \cdots$.)

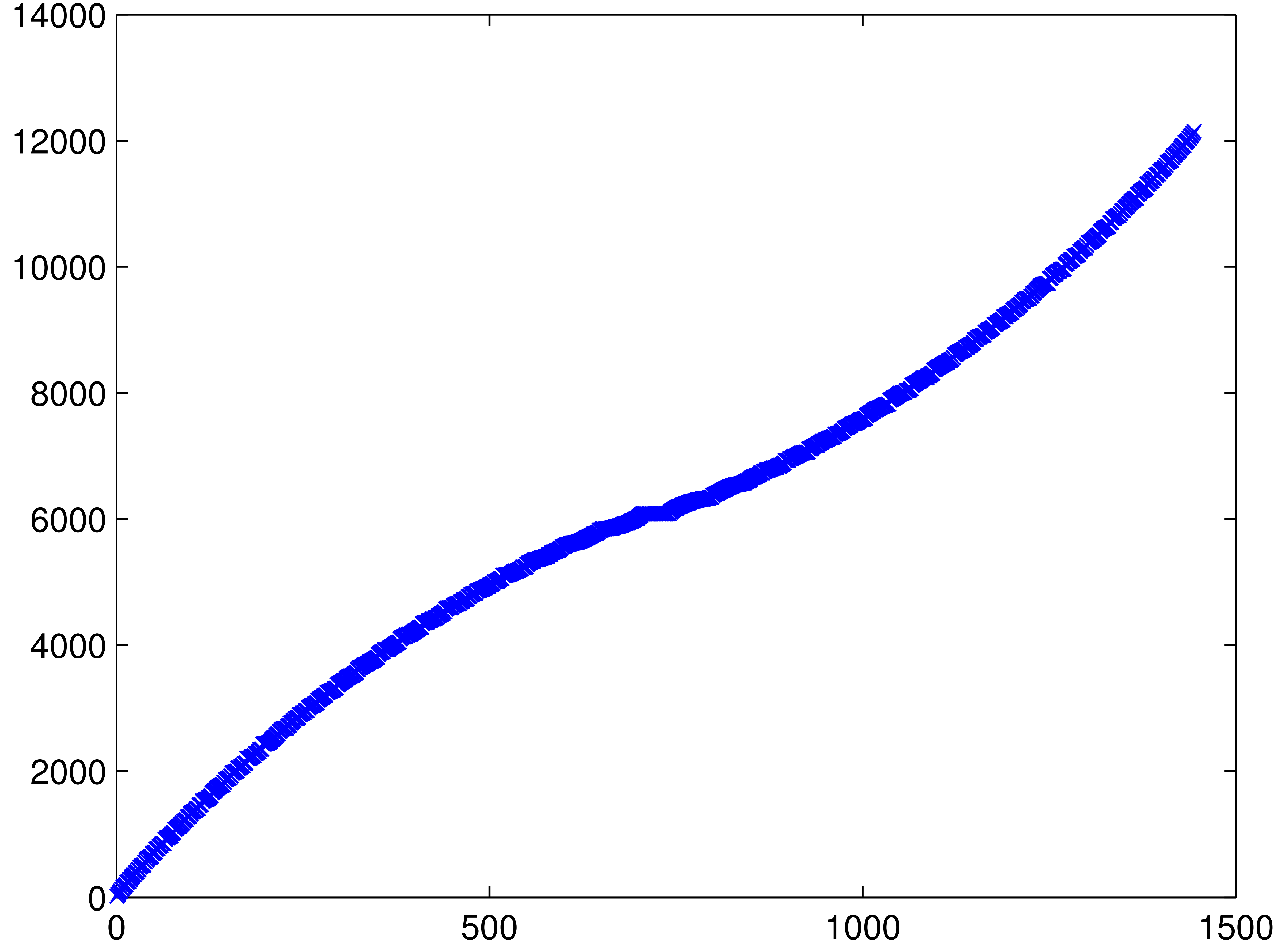
Question: What is the behavior of $\{\sigma_j\}_{j=1}^{N}$ as $N \to \infty$?

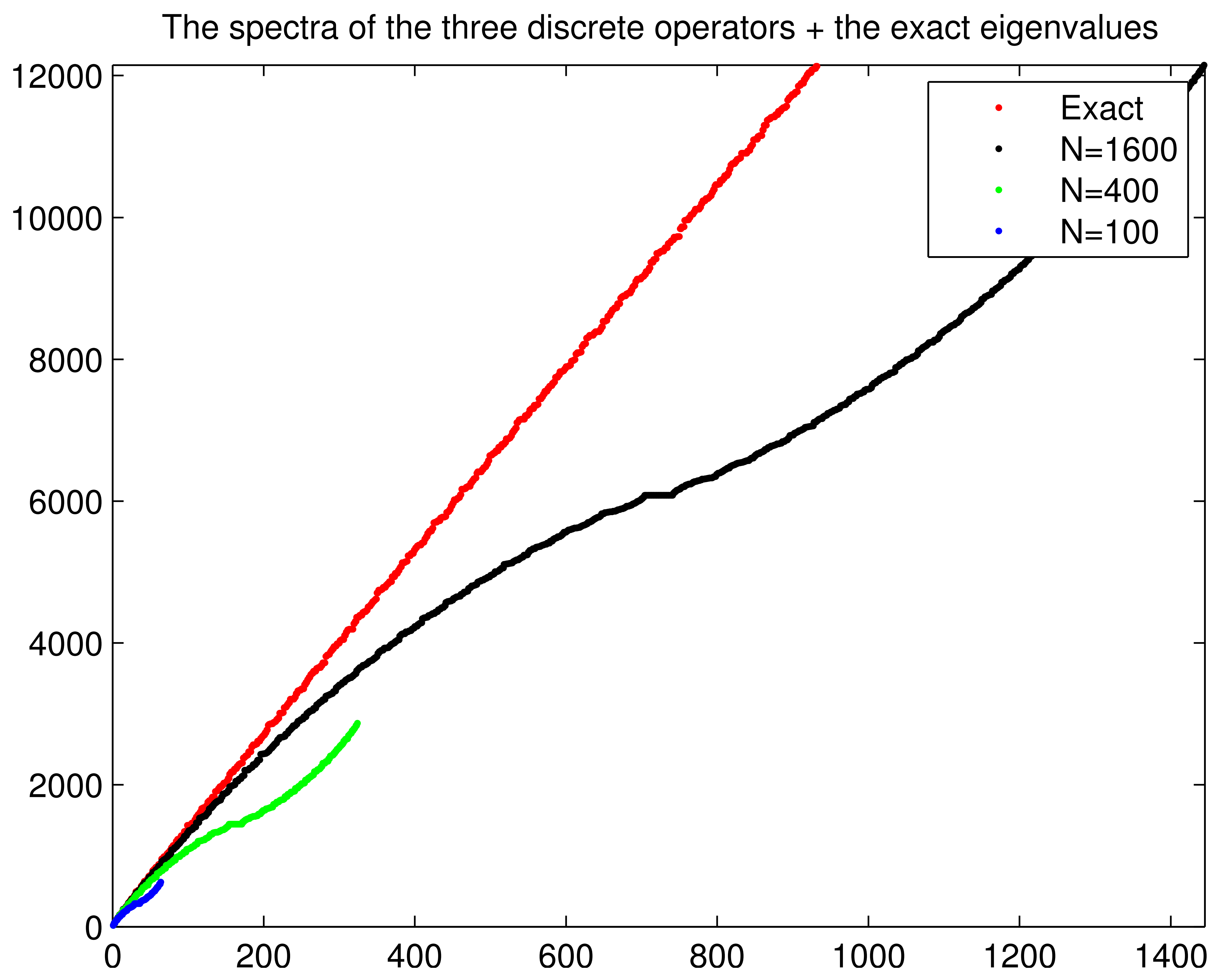
Singular values of A on 10 x 10 grid

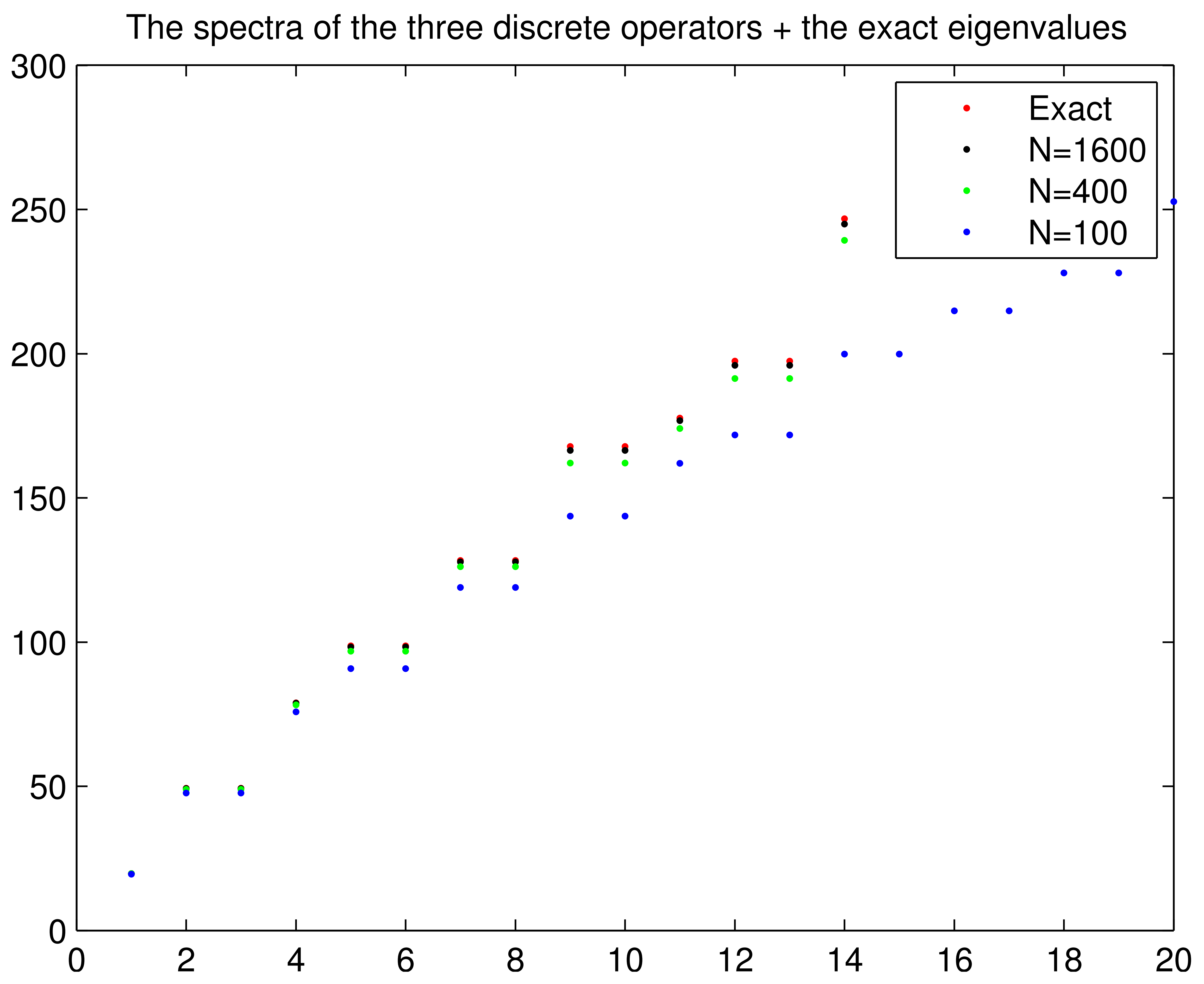



Singular values of A on 20 x 20 grid
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
350

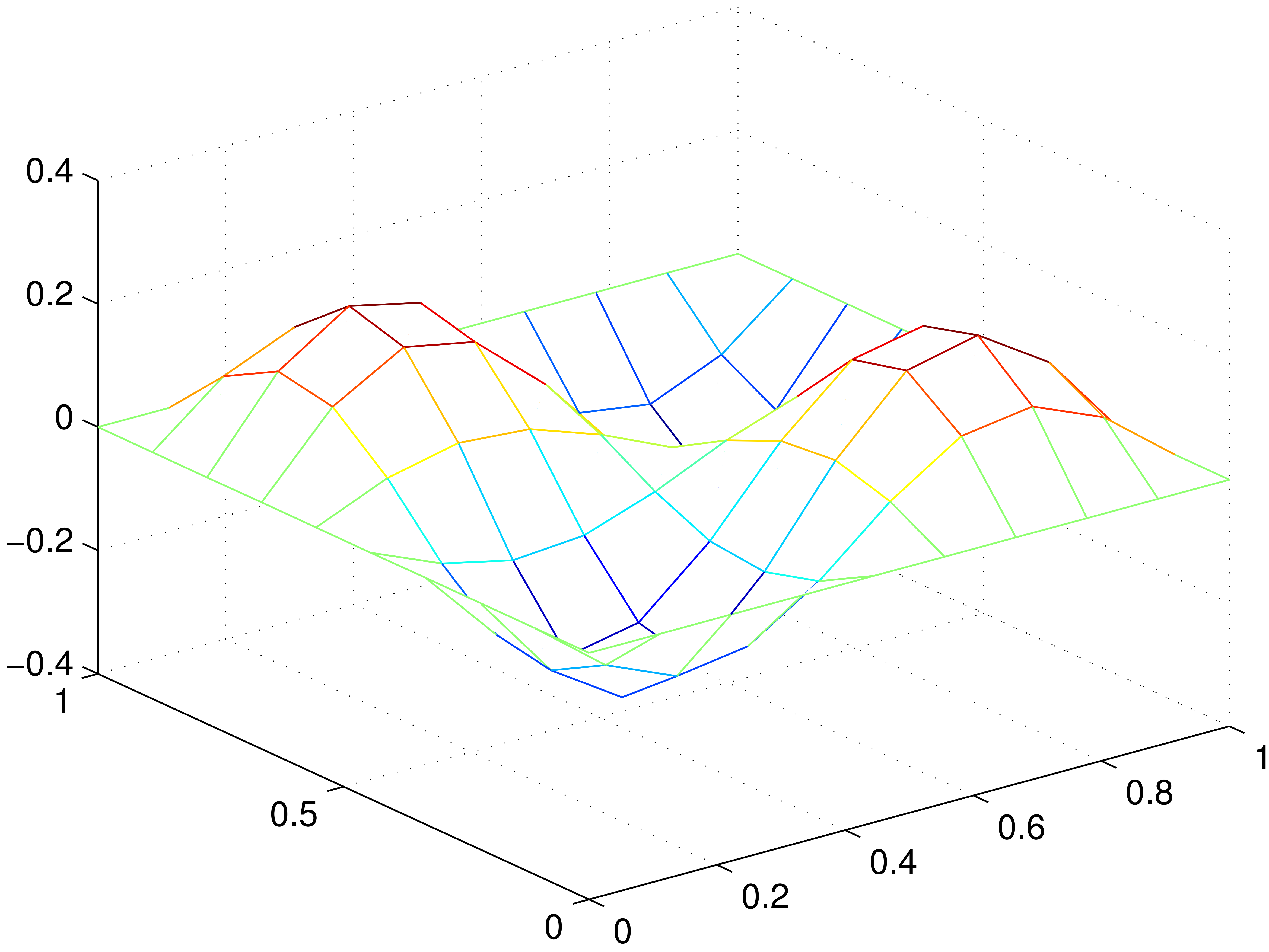
Singular values of A on 40 x 40 grid

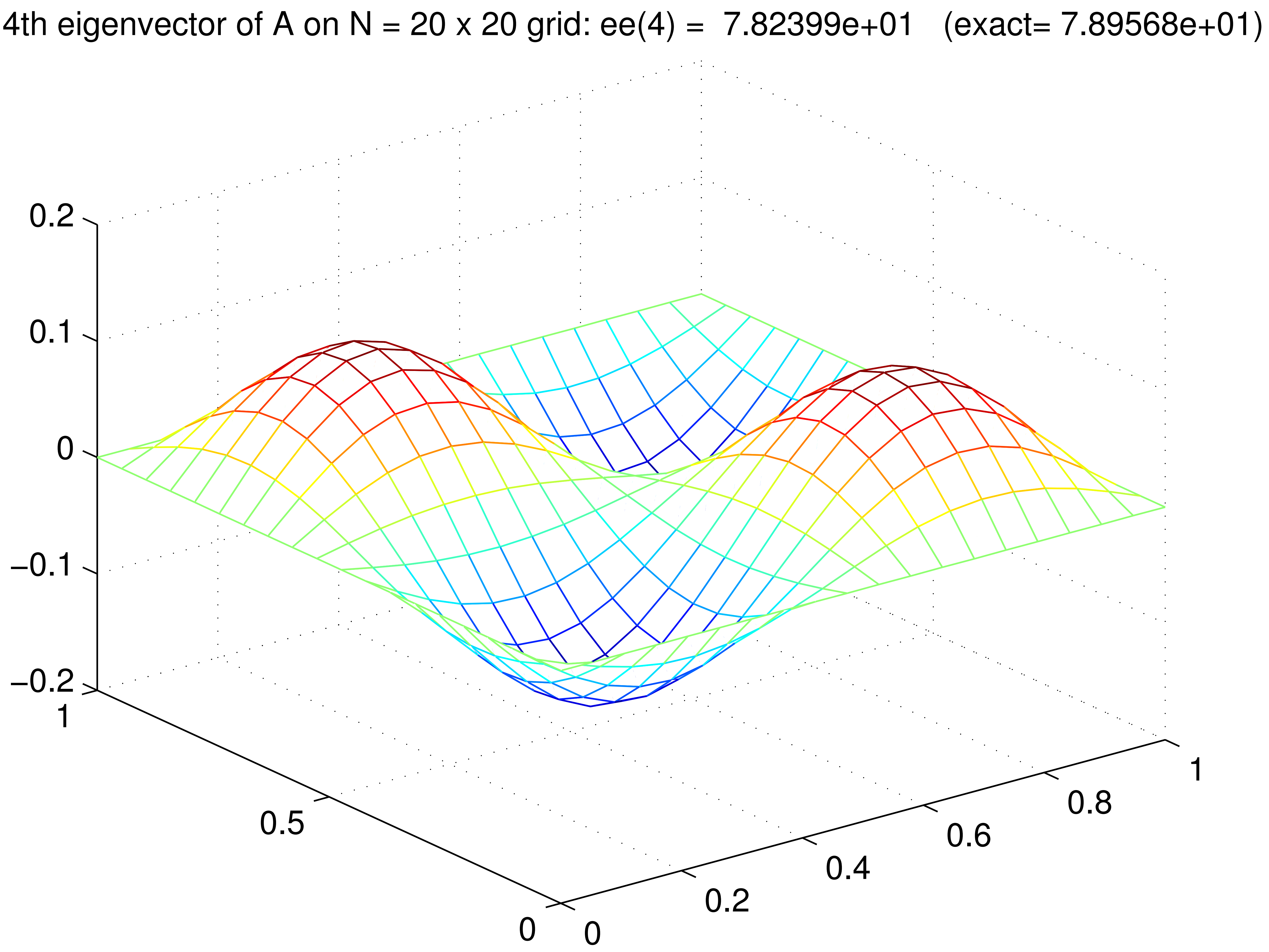



The spectra of the three discrete operators + the exact eigenvalues
12000
10000
8000
6000
4000
2000
0
0
200
400
600
800
1000
1200
1400
● Exact
● N=1600
● N=400
● N=100

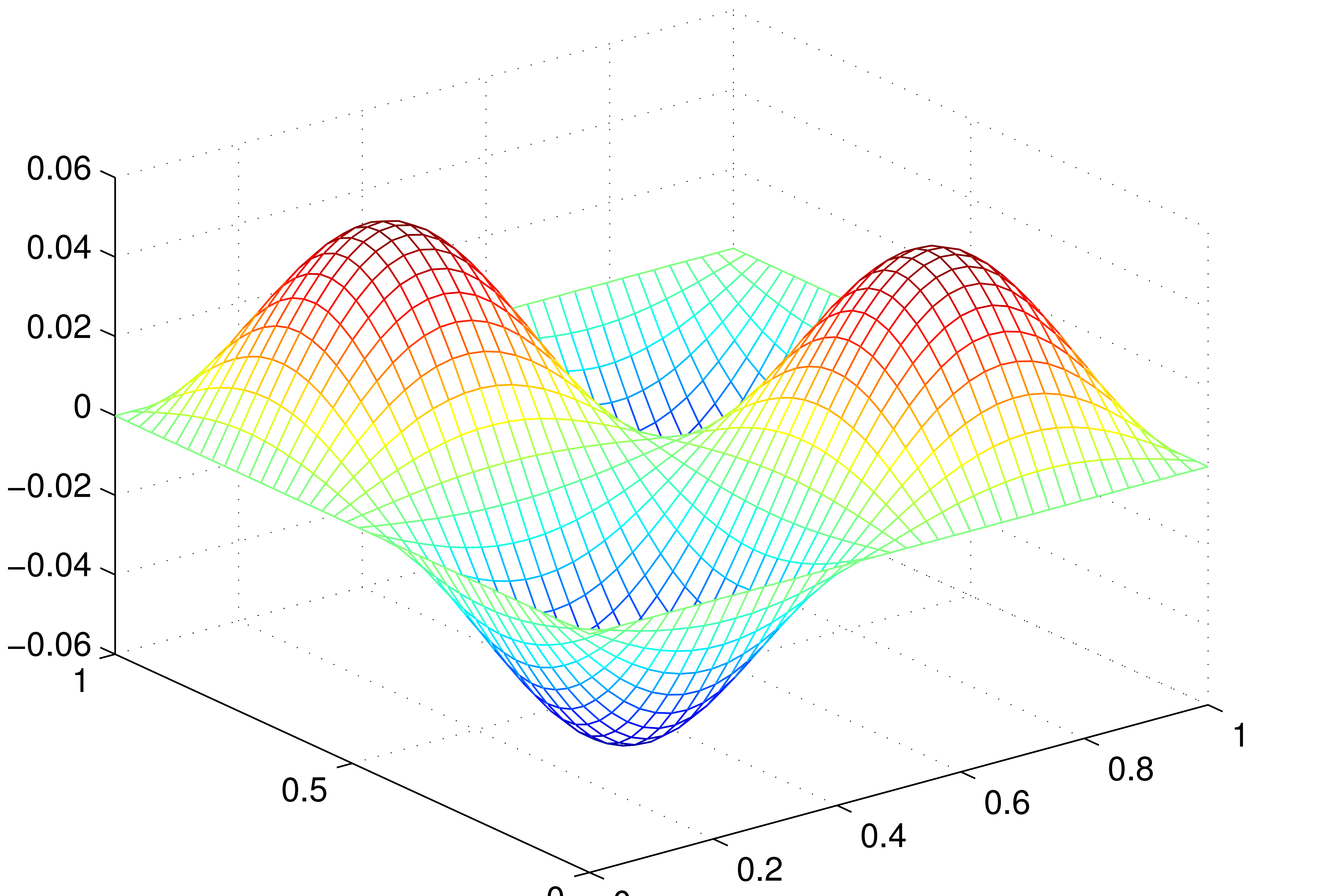

The spectra of the three discrete operators + the exact eigenvalues
Exact
N=1600
N=400
N=100
0
50
100
150
200
250
300
0
2
4
6
8
10
12
14
16
18
20

4th eigenvector of A on N = 10 x 10 grid: ee(4) = 7.58016e+01 (exact= 7.89568e+01)

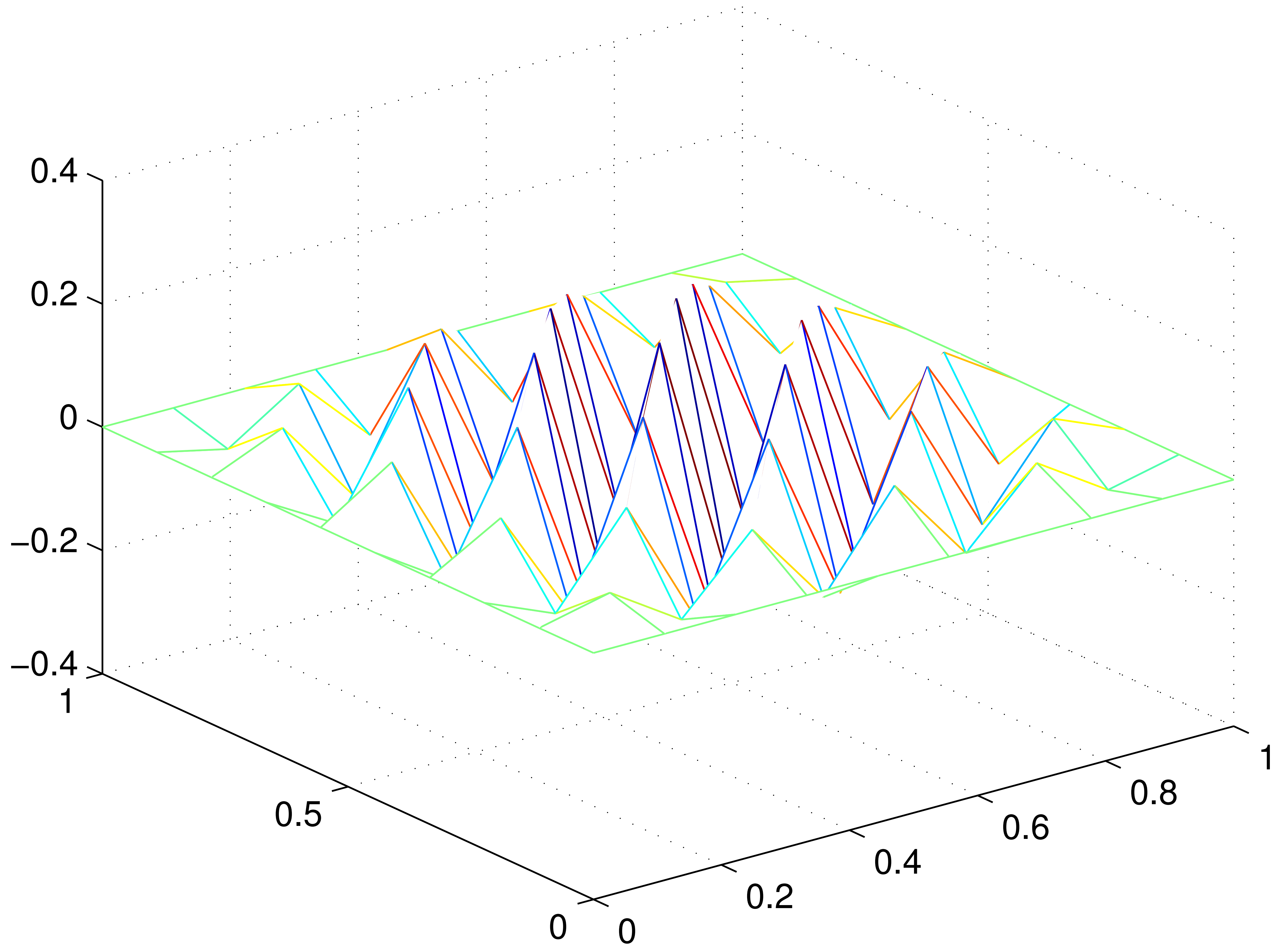



4th eigenvector of A on N = 20 x 20 grid: ee(4) = 7.82399e+01 (exact= 7.89568e+01)
0.2
0.1
0
−0.1
−0.2
1
0.5
0
0
0.2
0.4
0.6
0.8
1

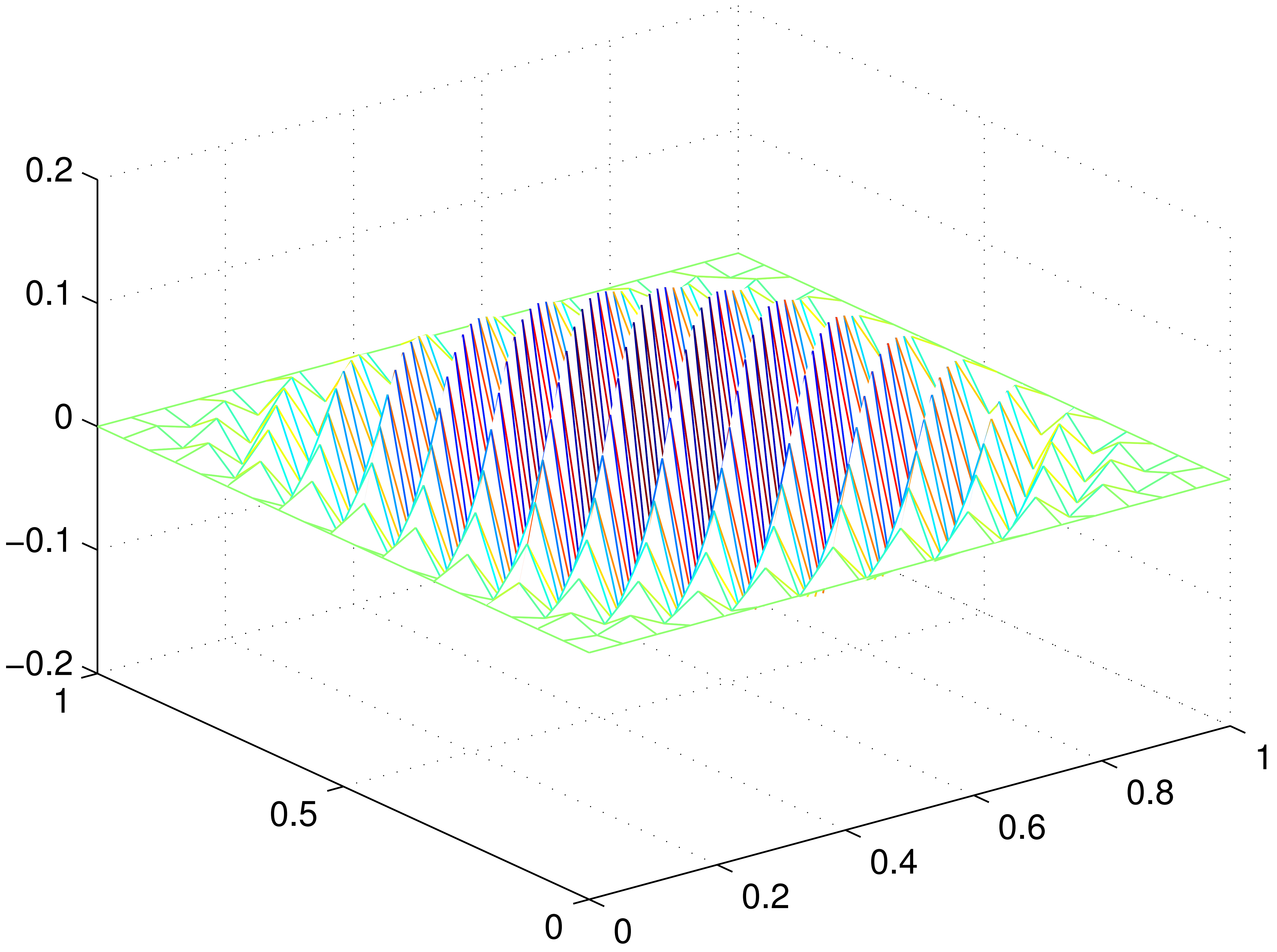
4th eigenvector of A on N = 40 x 40 grid: ee(4) = 7.87862e+01   (exact= 7.89568e+01)



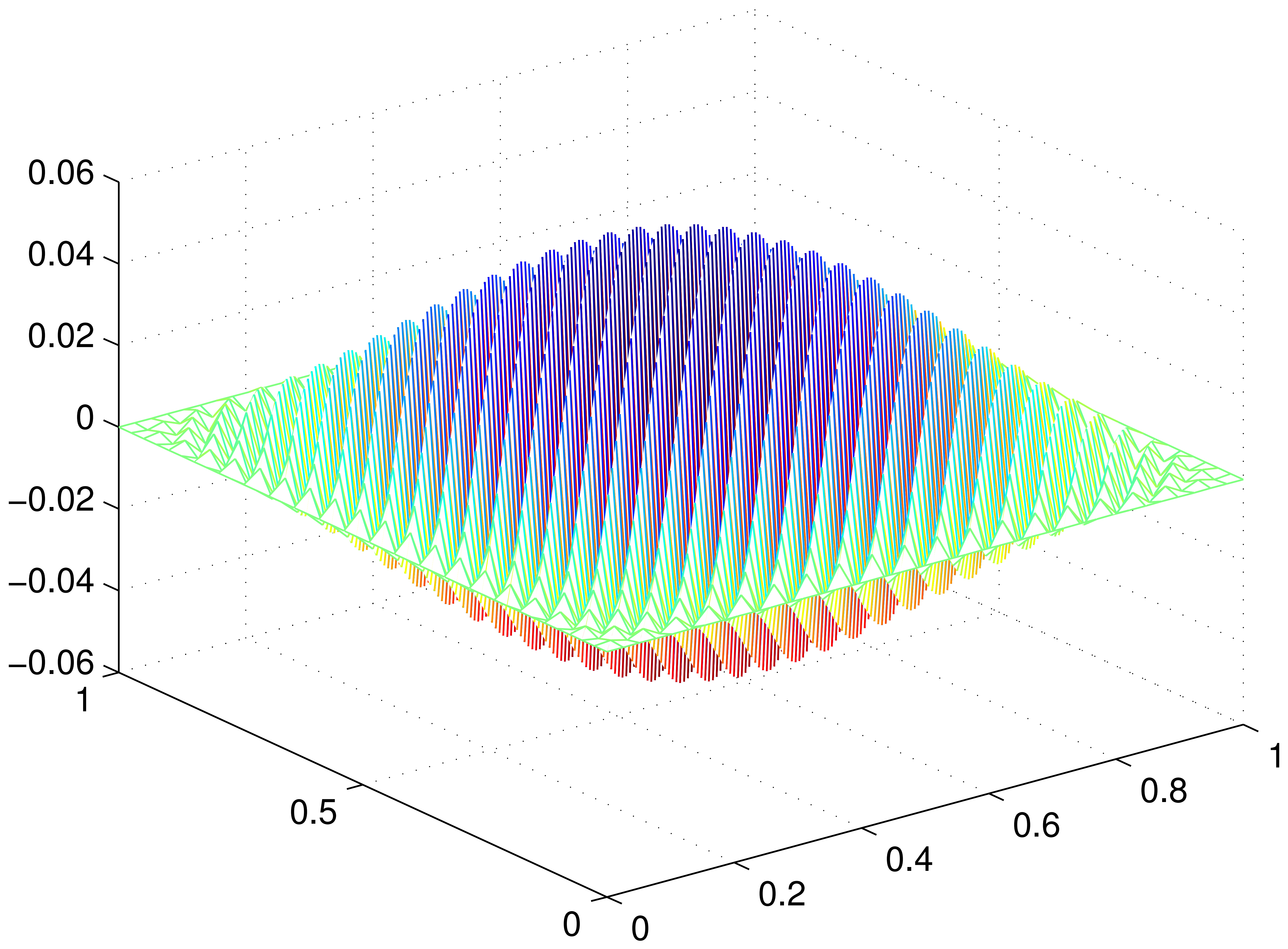
Last eigenvector of A on N = 10 x 10 grid: ee(end) = 6.28460e+02



Last eigenvector of A on N = 20 x 20 grid: ee(end) = 2.86831e+03



Last eigenvector of A on N = 40 x 40 grid: ee(end) = 1.21483e+04



Let $\mathbf{A}$ be the stiffness matrix of an $n \times n$ grid.

Let σ_j denote the jth smallest singular value of $\mathbf{A}$. (Recall backwards ordering.)

Claims:

- As $N \to \infty$, $\sigma_1(\mathbf{A})$ converges.
- As $N \to \infty$, $\sigma_N(\mathbf{A}) \sim h^{-2} \sim N$.
- In consequence, $\text{cond}(\mathbf{A}) = \frac{\sigma_N}{\sigma_1} \sim N$.

As N grows, we not only have the problem that $\mathbf{A}$ gets large, we also find that that vast majority of the "mass" in the matrix represents garbage that was introduced by the discretization. Good pre-conditioners, multigrid, etc, can to some extent ameliorate this problem.

The *inverse* of $\mathbf{A}$ is much nicer. Now the eigenmodes that are relevant (and that have any accuracy) are the dominant ones.

Very brief review of fast methods for solving $\mathbf{Ax} = \mathbf{b}$

- *Brute force LU factorization:* With the right pivots, complexity of the build stage is $O(N^{1.5})$ in 2D and $O(N^2)$ in 3D. Very stable and accurate. The solve stage has complexity $O(N \log N)$ in 2D and $O(N^{4/3})$ in 3D.

- *Iterative methods — conjugate gradients, GMRES, etc:*
  Each matvec is cheap ($O(N)$ cost) so total complexity is $O(N_{\rm iter} \times N)$.
  Convergence is a huge problem — $N_{\rm iter}$ can be large.
  Standard remedy is to precondition: Solve $\mathbf{PAx} = \mathbf{Pb}$, where $\mathbf{P} \approx \mathbf{A}^{-1}$.
  How to find such a **P** ... ?

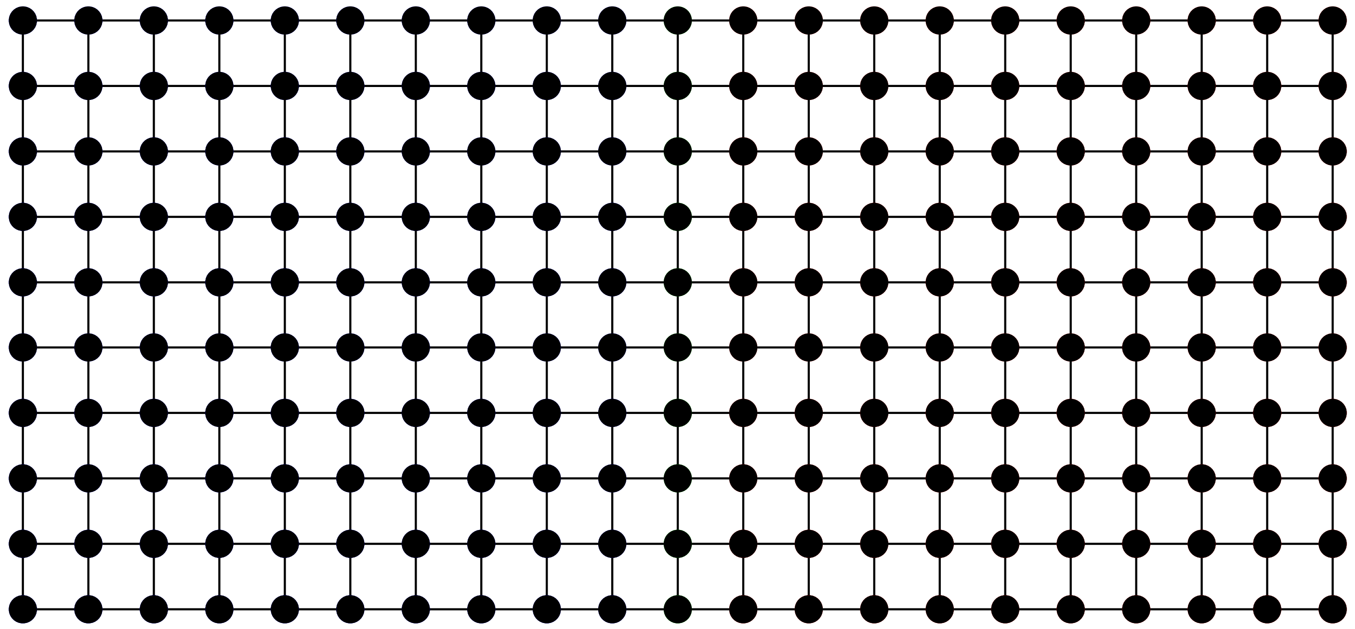
- *Multigrid:* Can be viewed as a variation of iterative methods but with the twist that **A** is represented on a hierarchy of grids (in a “multiresolution representation”).
  Then **A** is well-conditioned on each grid and convergence is very fast.
  Finding the right “projection” and “prolongation” operators can be dicey.

- *FFT:* If **A** represents a constant coefficient operator in a simple geometry (e.g. rectangle), then Fourier methods perform extremely well; but this only works in specialized environments (you can do better in these...). In other environments, the FFT can serve as a pre-conditioner.

Nested dissection (or, “how to choose the pivots in the LU decomposition”)

ND is a well-known “divide-and-conquer” technique, due to George (1973).

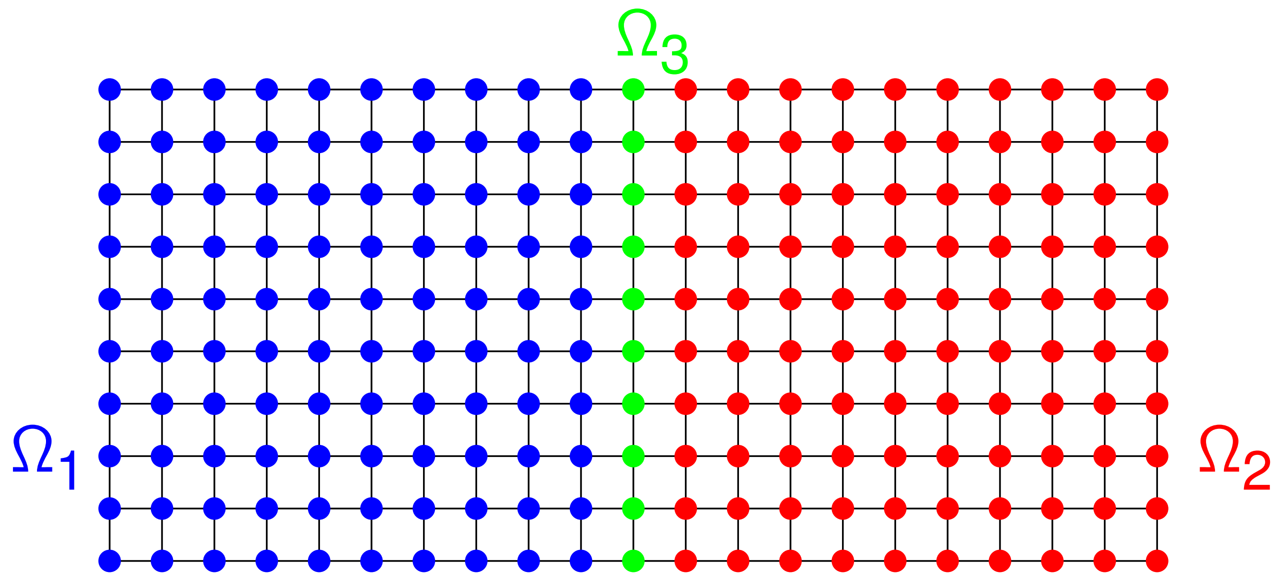
To illustrate the idea, consider a rectangular grid with $N = (2n+1) \times n$ gridpoints:



We seek to compute an LU-factorization of the $N \times N$ coefficient matrix $\mathbf{A}$.

Important: $\mathbf{A}$ has a lot of sparsity.

Let us divide the domain into three pieces:

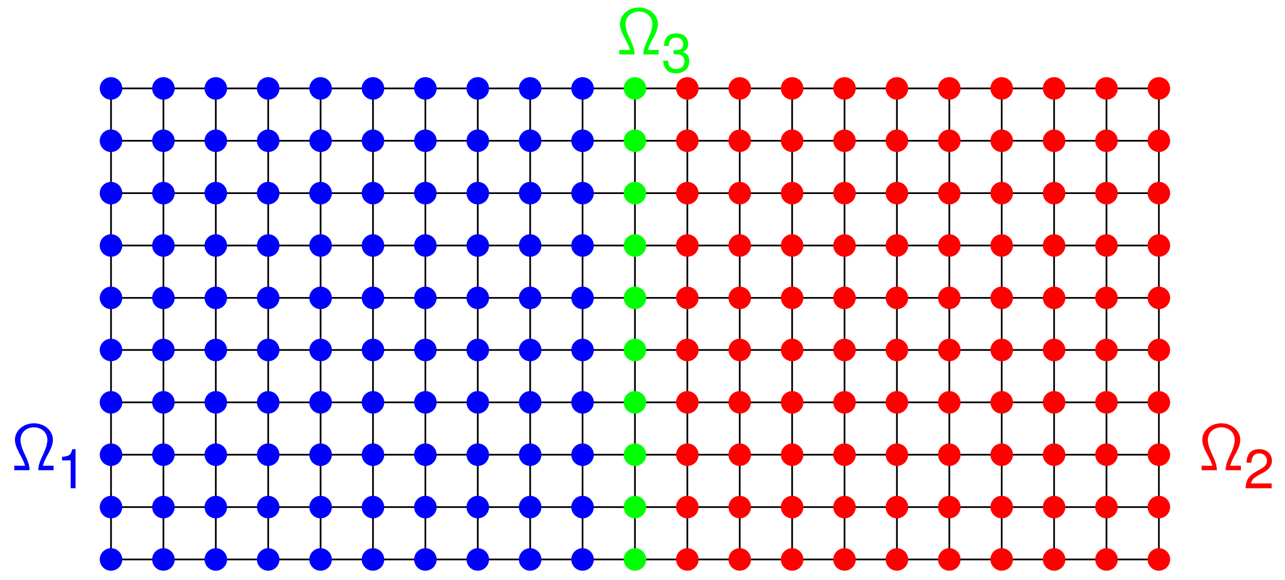


Note that there are no connections between nodes in Ω_1 and Ω_2.

In consequence, the non-zero blocks of the coefficient matrix are:

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

Now $\mathbf{A}_{13}$, $\mathbf{A}_{31}^{t}$, $\mathbf{A}_{23}$, and $\mathbf{A}_{32}^{t}$ have $n = O(N^{0.5})$ columns.



Recall that the coefficient matrix is tessellated as

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

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

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

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 the Schur complement $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$. $\sim \sqrt{N} \times \sqrt{N}$

Notice the obvious recursion!

Instead of *inverting* the matrix, one could construct its *LU-factorization* via an analogous process. This is slightly cheaper, and can be more numerically stable. Recall that

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

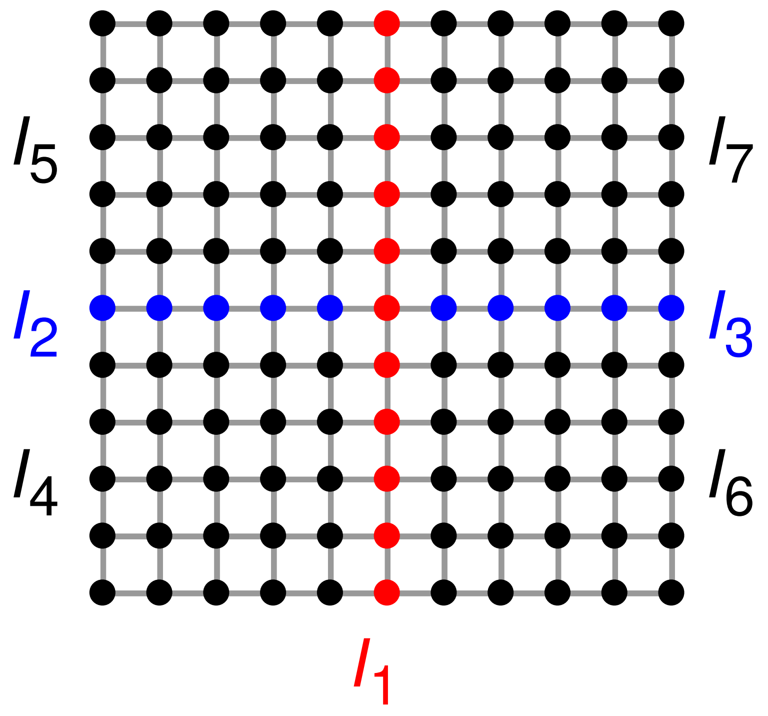
Now suppose that we have factorizations $\mathbf{A}_{11} = \mathbf{L}_{11}\mathbf{U}_{11}$ and $\mathbf{A}_{22} = \mathbf{L}_{22}\mathbf{U}_{22}$. Then

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

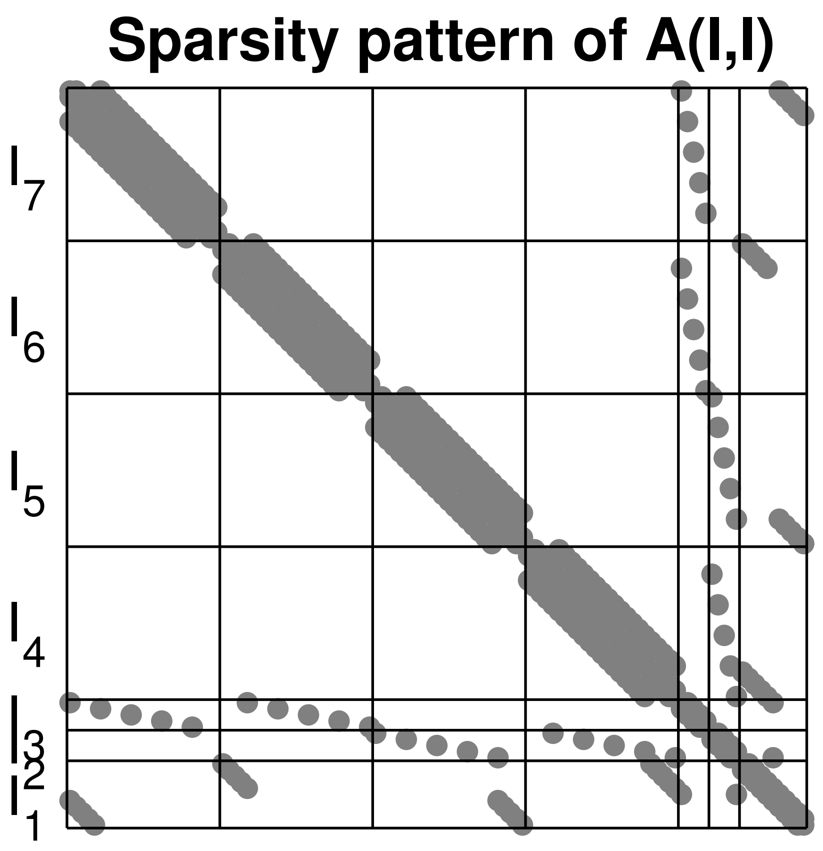
So in order to compute the LU factorization of $\mathbf{A}$, we need to:

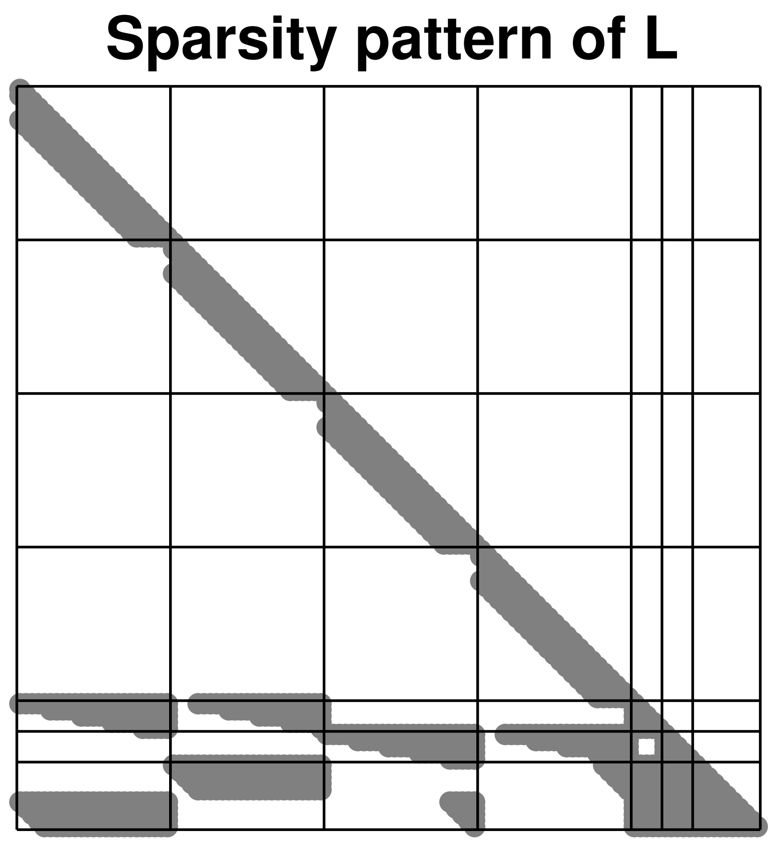
- Compute the factorization $\mathbf{A}_{11} = \mathbf{L}_{11}\,\mathbf{U}_{11}$. *size* $\sim N/2 \times N/2$
- Compute the factorization $\mathbf{A}_{22} = \mathbf{L}_{22}\,\mathbf{U}_{22}$. *size* $\sim N/2 \times N/2$
- Compute the Schur complement $\mathbf{S}_{33} = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{U}_{11}^{-1}\mathbf{L}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{U}_{22}^{-1}\mathbf{L}_{22}^{-1}\mathbf{A}_{23}$.
  Then compute the factorization $\mathbf{S}_{33} = \mathbf{L}_{33}\,\mathbf{U}_{33}$. *size* $\sim \sqrt{N} \times \sqrt{N}$

Next let us split the grid into 7 pieces:



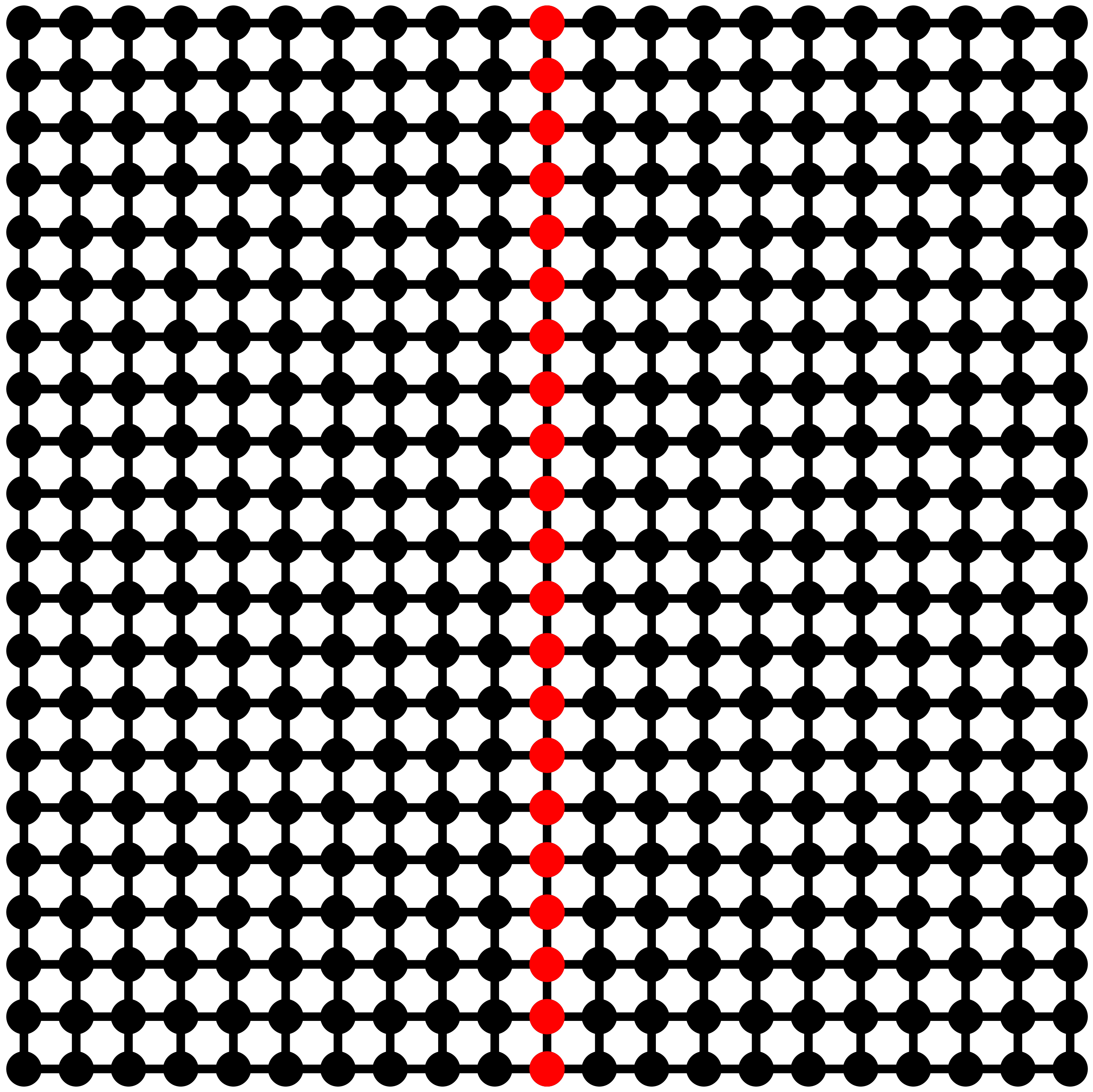
With $I = [I_7, I_6, I_5, I_4, I_3, I_2, I_1]$, the sparsity patterns in $\mathbf{A}(I, I) = \mathbf{LU}$ are:





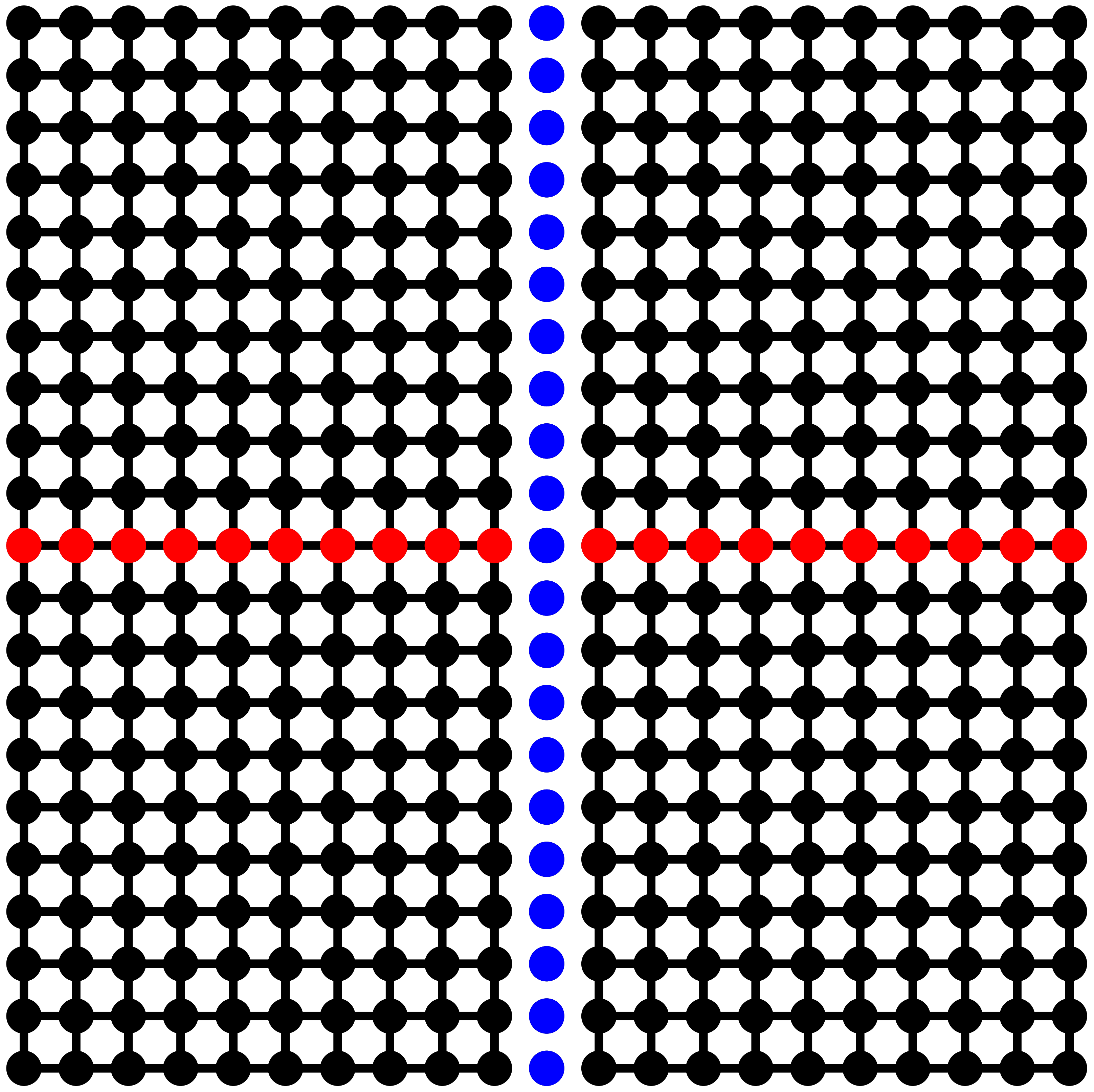
Geometry processing — no computations performed

Nested dissection – level 0



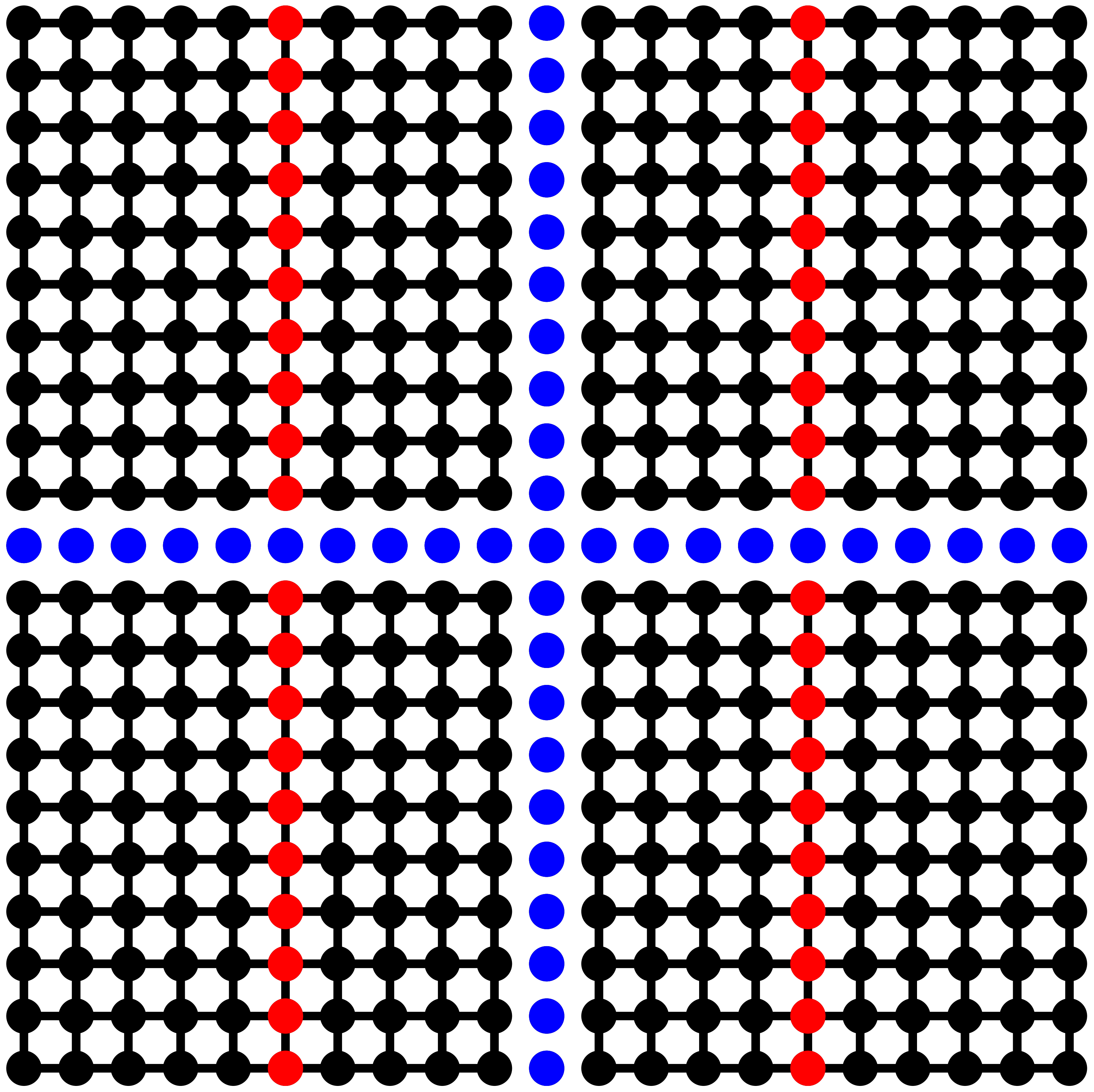
Geometry processing — no computations performed

Nested dissection – level 1



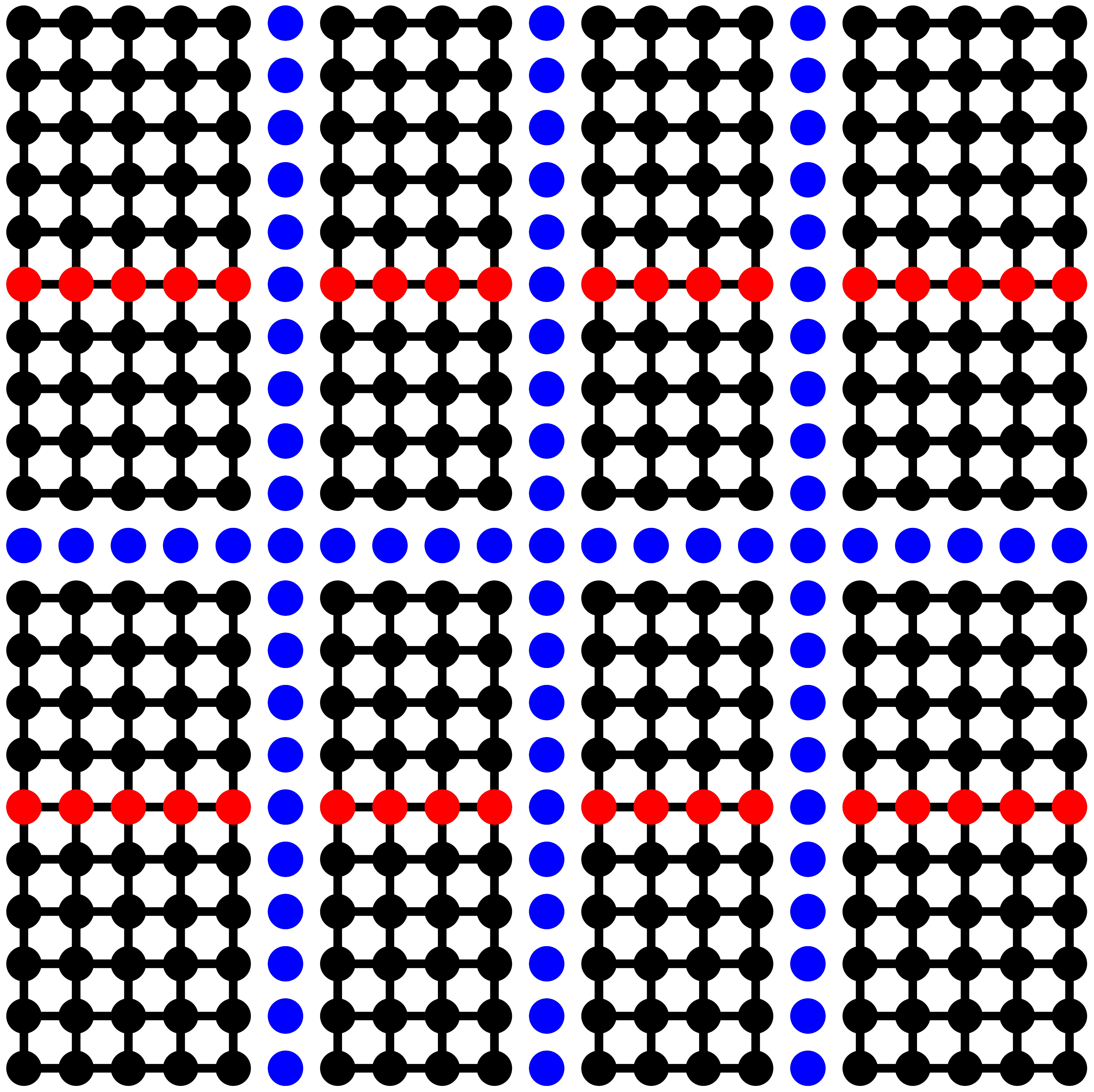
Geometry processing — no computations performed

Nested dissection – level 2



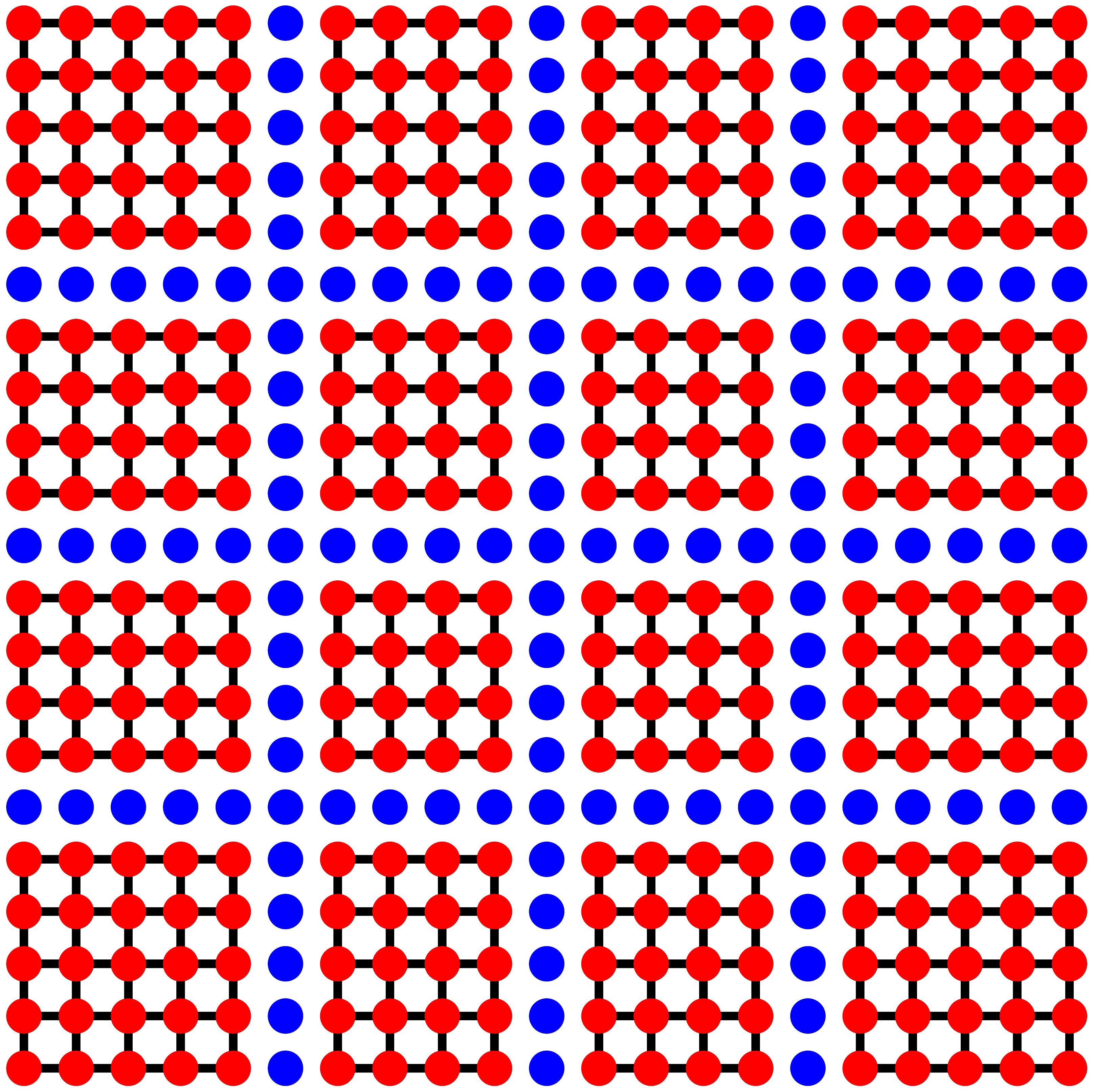
Geometry processing — no computations performed

Nested dissection – level 3

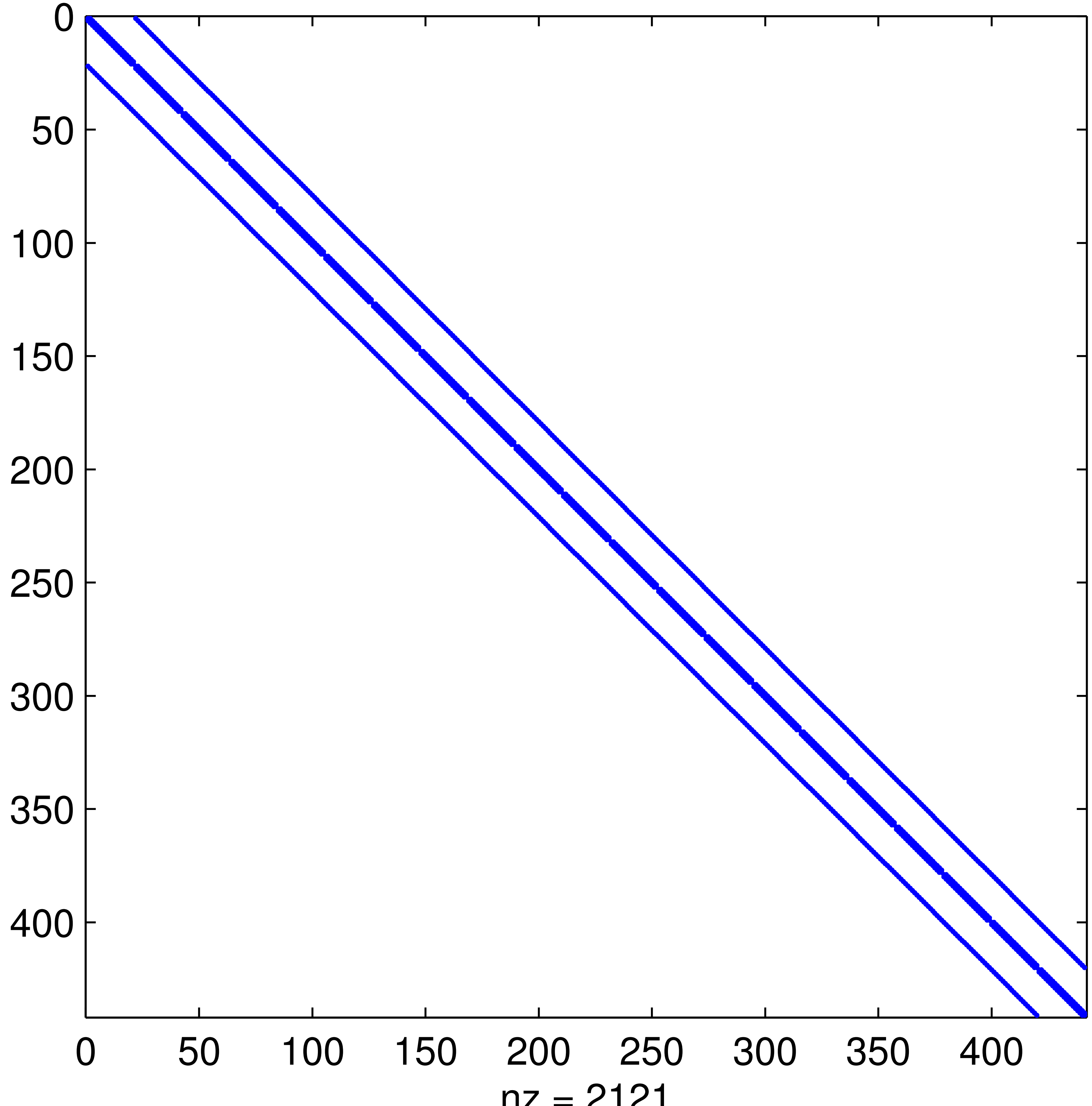


Geometry processing — no computations performed

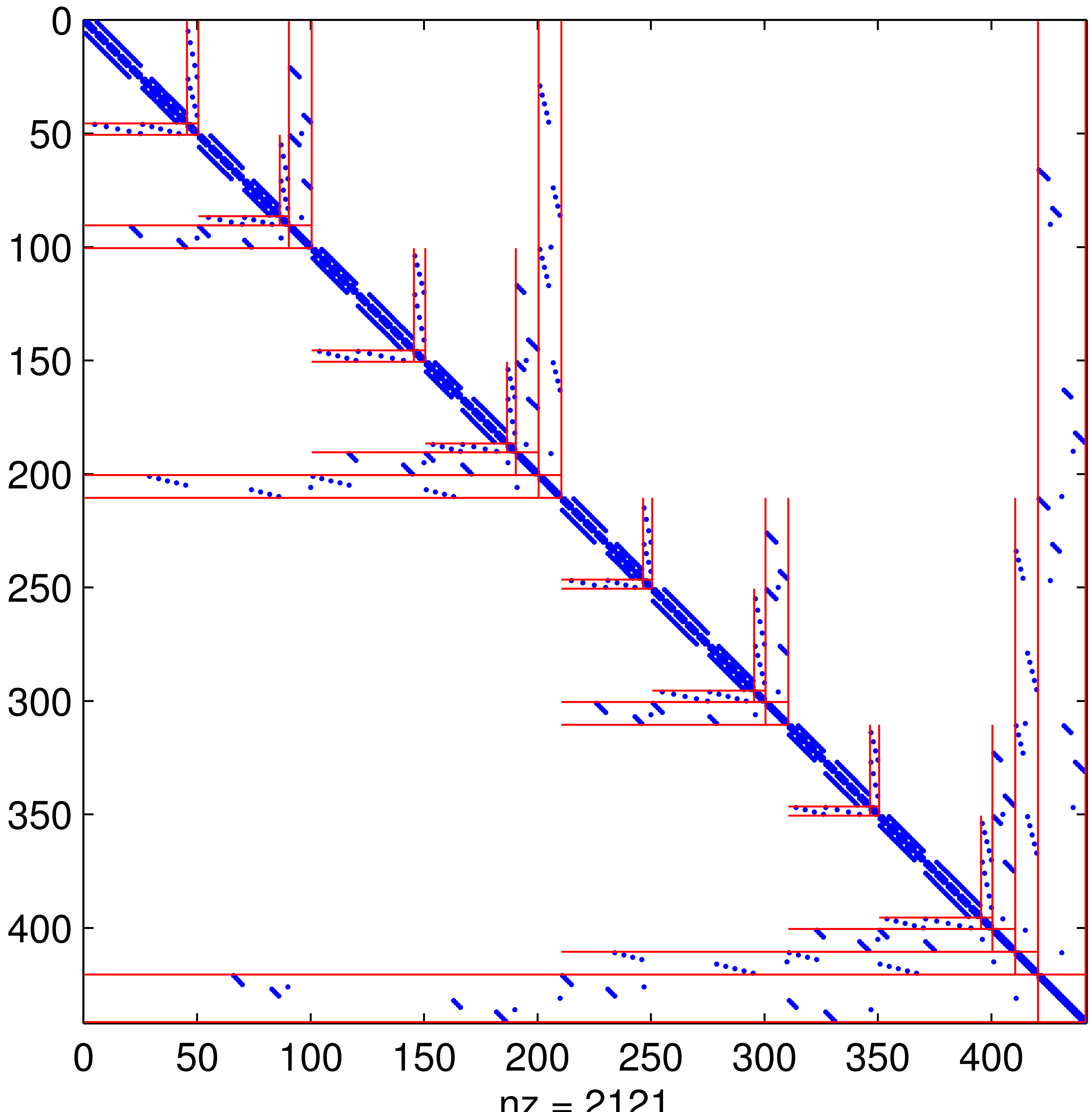
Nested dissection – level 4



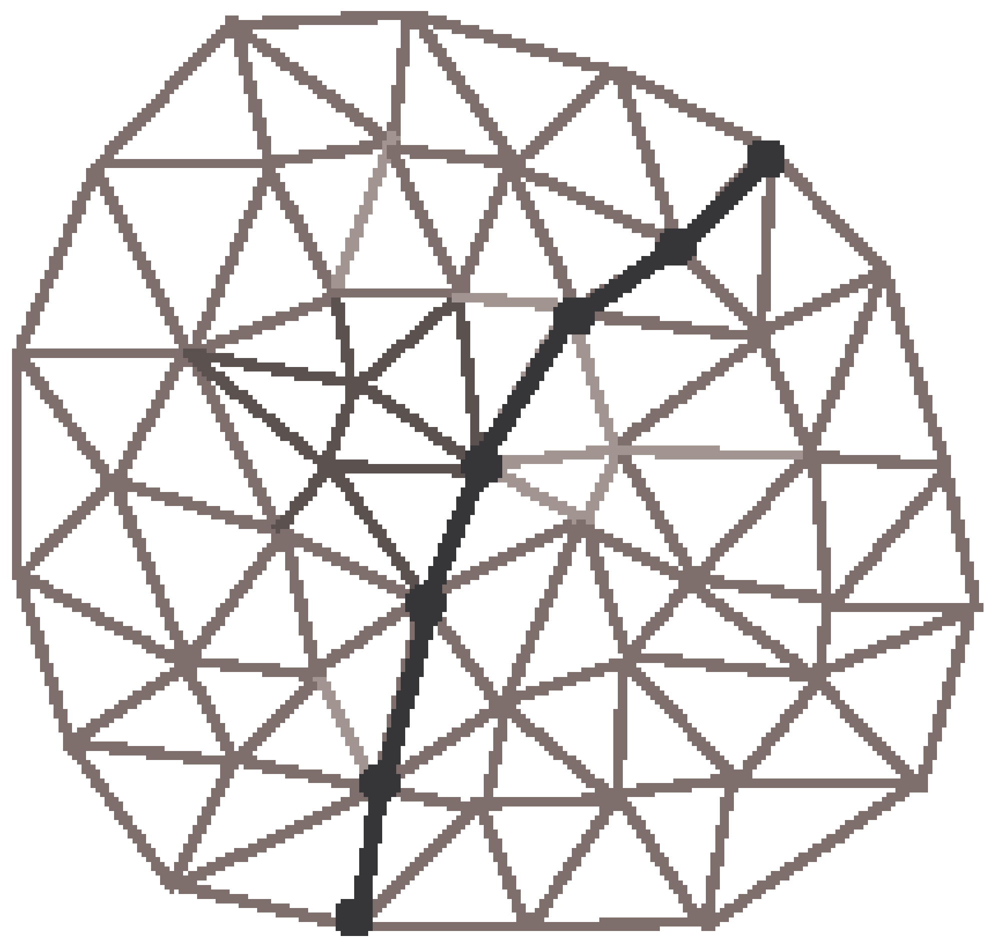
Sparsity pattern of **A** with column wise grid ordering.



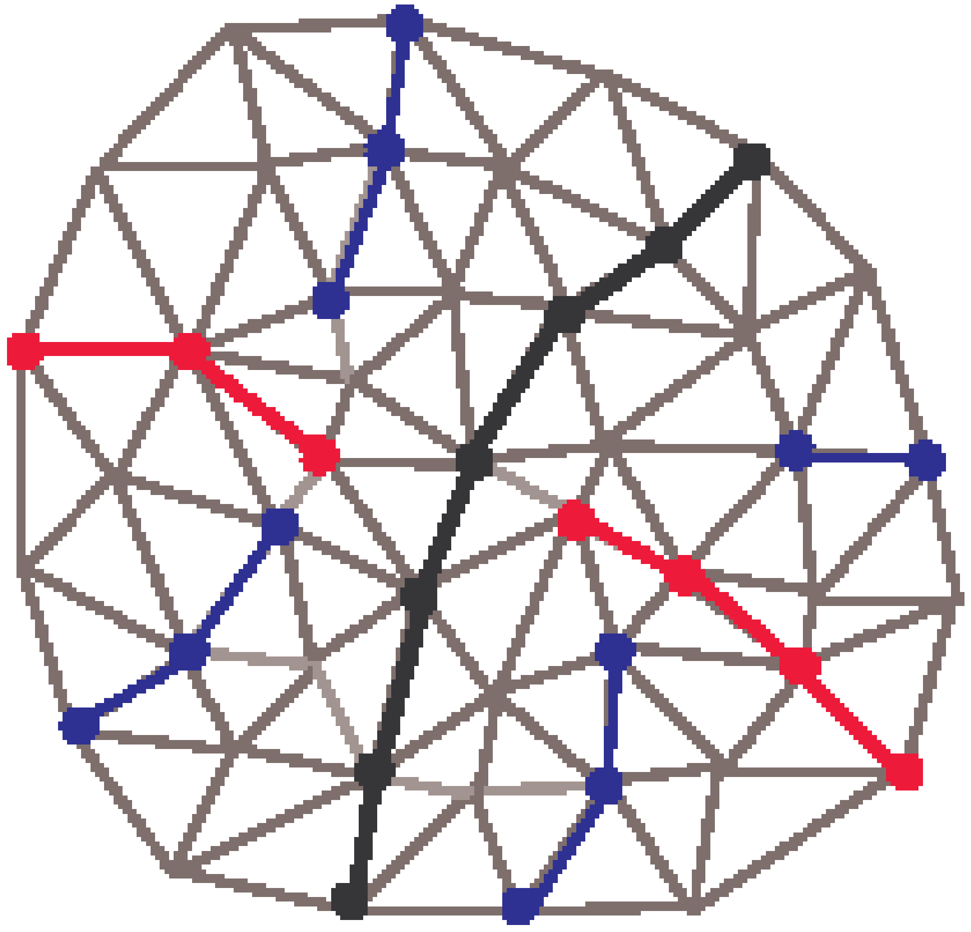
Sparsity pattern of **A** with nested dissection ordering.



Finding the best nested dissection ordering can be a challenge in real applications ...

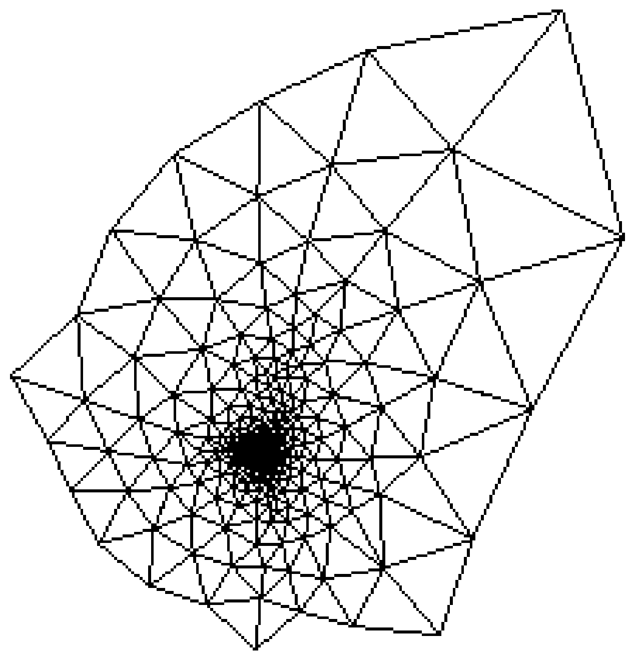


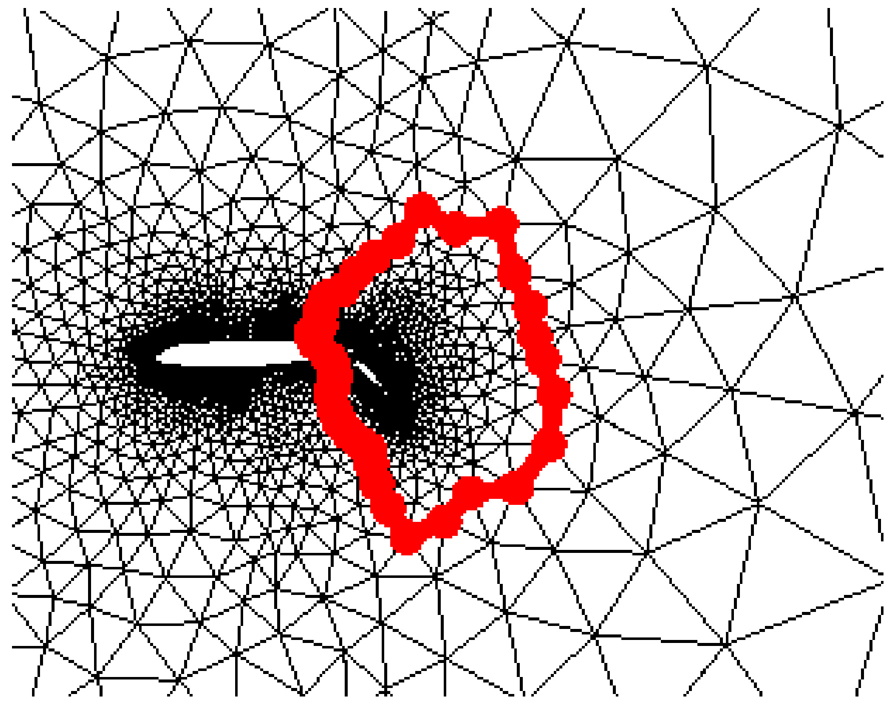
(i) One level partition

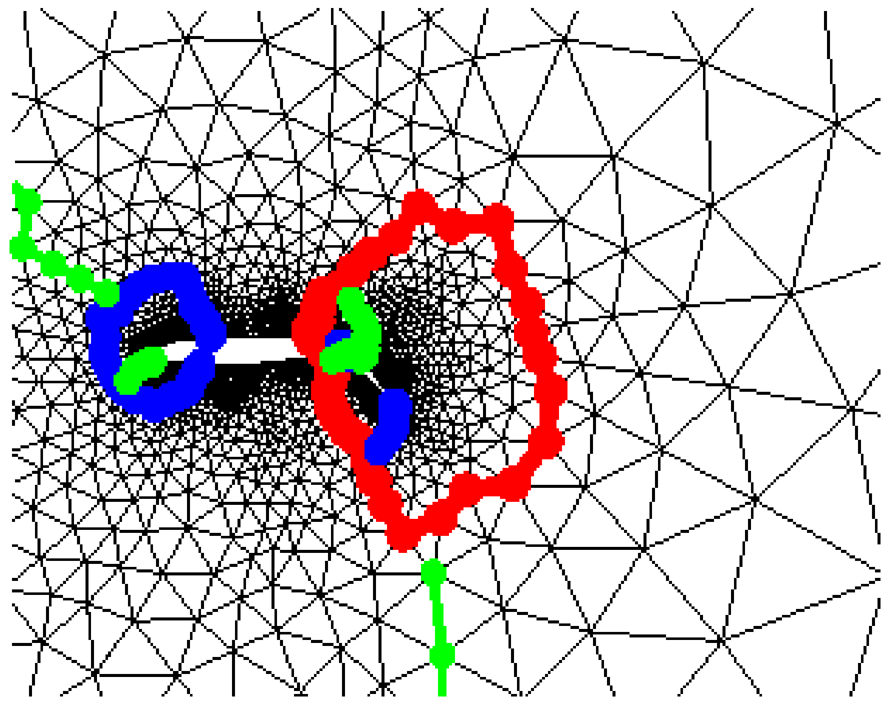


(ii) More partitions

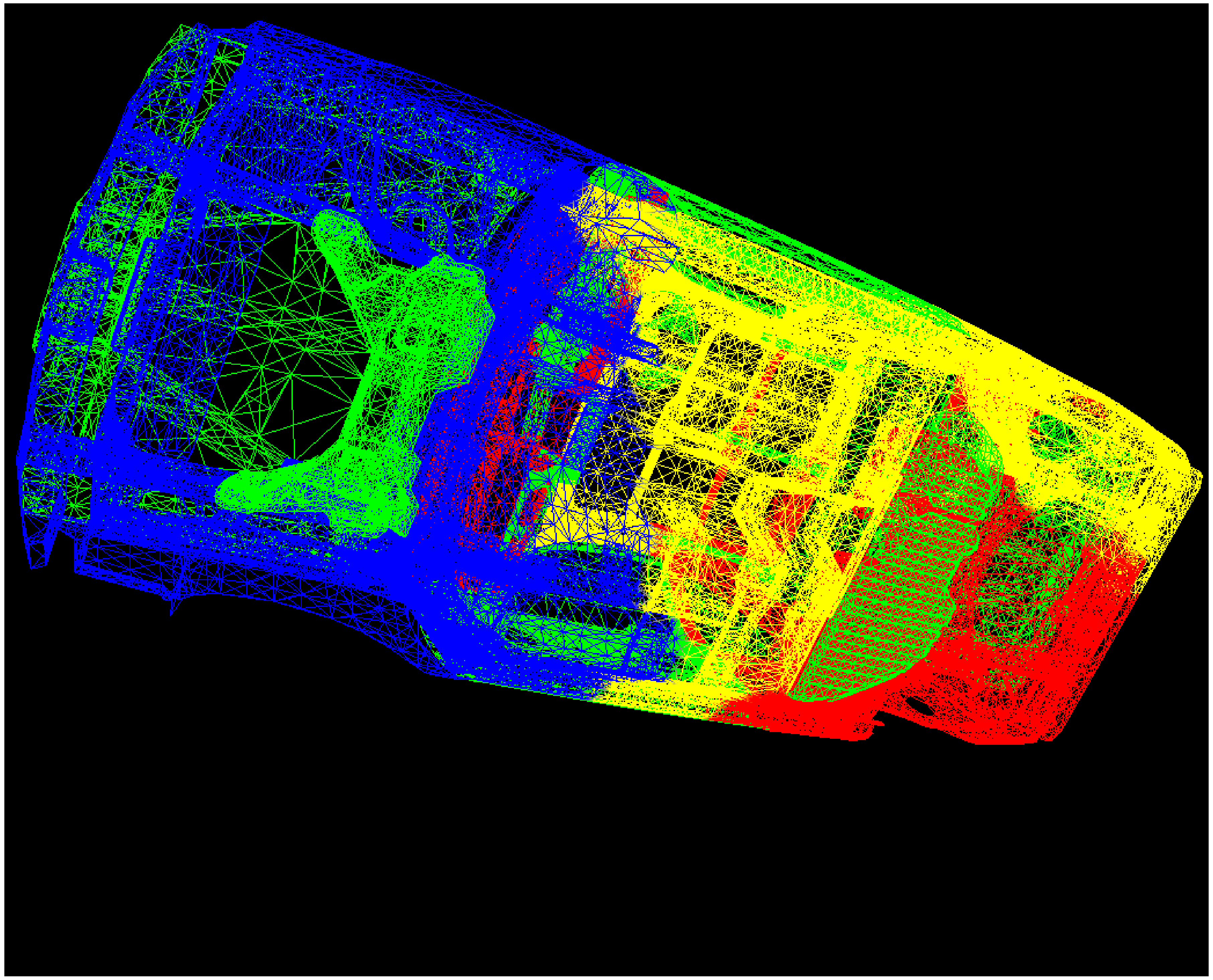
From: “Superfast Multifrontal Method for Large Structured Linear Systems of Equations.” Jianlin Xia, Shivkumar Chandrasekaran, Xiaoye S. Li. *SIAM J. Matrix Analysis Applications,* 2009; DOI:10.1137/09074543X


(i) *Original graph*


(ii) *Top level separator (zoomed in)*


(iii) *Three levels of separators*

From: “Robust and Efficient Multifrontal Solver for Large Discretized PDEs.” Jianlin Xia. In *High-Performance Scientific Computing,* 2012; DOI:10.1007/978-1-4471-2437-5_10



From: http://www.labri.fr/perso/pelegrin/scotch/

However, the actual LU factorization is simple, at least at a “pseudo-code” level.

```
LU FACTORIZATION VIA NESTED DISSECTION.

Input: An $N \times N$ matrix $\mathbf{A}$, and a $2 \times N$ array $\mathbf{X}$ holding all grid points.
Output: An index vector $I$ and lower/upper triangular matrices $\mathbf{L}$ and $\mathbf{U}$ such that $\mathbf{A}(I, I) = \mathbf{L}\mathbf{U}$.
----------------------------------------------------------------
Find a nested dissection ordering of the mesh $\{I_\tau\}_{\tau=1}^{M} = \texttt{ND\_ordering}(\mathbf{A}, \mathbf{X})$;
$I = [I_M,\, I_{M-1},\, \dots,\, I_2,\, I_1]$;
$\mathbf{L} = \mathbf{I}$; $\mathbf{U} = \mathbf{I}$;
for $\tau = M,\, M-1,\, M-2,\, \dots,\, 2$
    $[\tilde{\mathbf{L}}, \tilde{\mathbf{U}}] = \texttt{lu}(\mathbf{A}(I_\tau, I_\tau))$.
    $\mathbf{L}(I_\tau, I_\tau) = \tilde{\mathbf{L}}$.
    $\mathbf{L}(I_\tau^{\rm c}, I_\tau) = \mathbf{A}(I_\tau^{\rm c}, I_\tau)\tilde{\mathbf{U}}^{-1}$.
    $\mathbf{U}(I_\tau, I_\tau) = \tilde{\mathbf{U}}$.
    $\mathbf{U}(I_\tau, I_\tau^{\rm c}) = \tilde{\mathbf{L}}^{-1}\mathbf{A}(I_\tau, I_\tau^{\rm c})$.
    $\mathbf{A}(I_\tau^{\rm c}, I_\tau^{\rm c}) = \mathbf{A}(I_\tau^{\rm c}, I_\tau^{\rm c}) - \mathbf{L}(I_\tau^{\rm c}, I_\tau)\mathbf{U}(I_\tau, I_\tau^{\rm c})$.
end for
$[\tilde{\mathbf{L}}, \tilde{\mathbf{U}}] = \texttt{lu}(\mathbf{A}(I_1, I_1))$.
$\mathbf{L}(I_1, I_1) = \tilde{\mathbf{L}}$.
$\mathbf{U}(I_1, I_1) = \tilde{\mathbf{U}}$.
```

LU factorization via nested dissection. Observe that the matrices $\mathbf{A}(I_\tau^{\rm c}, I_\tau)$ and $\mathbf{A}(I_\tau, I_\tau^{\rm c})$ are for the most part very sparse, and not many elements are actually updated in each computation.

Actually getting high performance in the sparse matrix arithmetic is a different matter...

Asymptotic cost of nested dissection — *two dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the “divider” has $\sim N^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0:} \sim \left(N^{1/2}\right)^3 \sim N^{3/2}.$$

Asymptotic cost of nested dissection — *two dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the "divider" has $\sim N^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0: } \sim \left(N^{1/2}\right)^3 \sim N^{3/2}.$$

At the next finer level, there are 2 "dividers", each having $\sim (N/2)^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 1: } \sim 2\left((N/2)^{1/2}\right)^3 \sim 2^{-1/2} N^{3/2}.$$

Asymptotic cost of nested dissection — *two dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the "divider" has $\sim N^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0: } \sim \left(N^{1/2}\right)^3 \sim N^{3/2}.$$

At the next finer level, there are 2 "dividers", each having $\sim (N/2)^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 1: } \sim 2\left((N/2)^{1/2}\right)^3 \sim 2^{-1/2}N^{3/2}.$$

At the next finer level, there are 4 "dividers", each having $\sim (N/4)^{1/2}$ points...

Asymptotic cost of nested dissection — *two dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the “divider” has $\sim N^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0: } \sim \left(N^{1/2}\right)^3 \sim N^{3/2}.$$

At the next finer level, there are 2 “dividers”, each having $\sim (N/2)^{1/2}$ points and we need to perform dense inversion.

$$\text{cost of processing level 1: } \sim 2\left((N/2)^{1/2}\right)^3 \sim 2^{-1/2}\,N^{3/2}.$$

At the next finer level, there are 4 “dividers”, each having $\sim (N/4)^{1/2}$ points...

$$\begin{array}{llll}
\text{Cost of processing level 0:} & \sim\ 1\,((N/1)^{1/2})^3 & \sim & 1^{-1/2}\,N^{3/2} \\
\text{Cost of processing level 1:} & \sim\ 2\,((N/2)^{1/2})^3 & \sim & 2^{-1/2}\,N^{3/2} \\
\text{Cost of processing level 2:} & \sim\ 4\,((N/4)^{1/2})^3 & \sim & 4^{-1/2}\,N^{3/2} \\
\text{Cost of processing level 3:} & \sim\ 8\,((N/8)^{1/2})^3 & \sim & 8^{-1/2}\,N^{3/2} \\
\vdots & & \vdots & \vdots \\
\hline
\text{Total cost:} & & \sim & N^{3/2}
\end{array}$$

Asymptotic cost of nested dissection — *three dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the "divider" has $\sim N^{2/3}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0: } \sim \left(N^{2/3}\right)^3 \sim N^2.$$

Asymptotic cost of nested dissection — *three dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the "divider" has $\sim N^{2/3}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0: } \sim \left(N^{2/3}\right)^3 \sim N^2.$$

At the next finer level, there are 2 "dividers", each having $\sim (N/2)^{2/3}$ points and we need to perform dense inversion.

$$\text{cost of processing level 1: } \sim 2\left((N/2)^{2/3}\right)^3 \sim 2^{-1/3} N^2.$$

Asymptotic cost of nested dissection — *three dimensions*

Let N denote the total number of nodes in the domain.

At the top level, the "divider" has $\sim N^{2/3}$ points and we need to perform dense inversion.

$$\text{cost of processing level 0:} \sim \left(N^{2/3}\right)^3 \sim N^2.$$

At the next finer level, there are 2 "dividers", each having $\sim (N/2)^{2/3}$ points and we need to perform dense inversion.

$$\text{cost of processing level 1:} \sim 2\left((N/2)^{2/3}\right)^3 \sim 2^{-1/3}N^2.$$

At the next finer level, there are 4 "dividers", each having $\sim (N/4)^{1/2}$ points...

$$\begin{array}{lcrcr}
\text{Cost of processing level 0:} & \sim & 1\,((N/1)^{2/3})^3 & \sim & 1^{-1}\,N^{3/2} \\
\text{Cost of processing level 1:} & \sim & 2\,((N/2)^{2/3})^3 & \sim & 2^{-1}\,N^{3/2} \\
\text{Cost of processing level 2:} & \sim & 4\,((N/4)^{2/3})^3 & \sim & 4^{-1}\,N^{3/2} \\
\text{Cost of processing level 3:} & \sim & 8\,((N/8)^{2/3})^3 & \sim & 8^{-1}\,N^{3/2} \\
\vdots & & \vdots & & \vdots \\
\hline
\text{Total cost:} & & & \sim & N^2
\end{array}$$

To summarize, the costs of classical nested dissection are:

	Build stage	*Solve stage*	*Memory*
2D	$O(N^{3/2})$	$O(N \log N)$	$O(N \log N)$
3D	$O(N^2)$	$O(N^{4/3})$	$O(N^{4/3})$

The algorithm we have described is well-established, and goes under names such as "nested dissection methods" and "multifrontal methods".

Highly optimized software libraries are available, MUMPS, Pardiso, etc.
Note that while all examples shown here involves the simplest possible grids, there is no conceptual problem in handling more general grids. There are of course algorithmic issues, how do you find the best cuts, etc, but very good algorithms exist.

These codes are in 2D very efficient for moderate N (say up to $N \sim 10^7$).

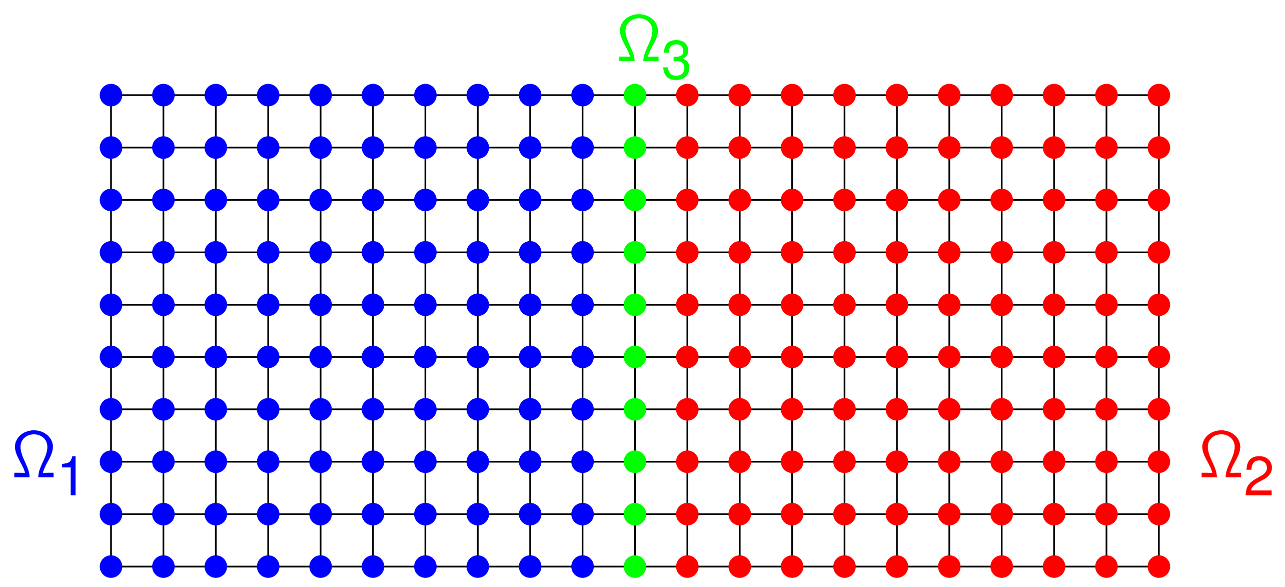
In 3D, performance is not quite what one could wish — parallelization is challenging.

Also, high order stencils cause much worse practical performance.

References: Several books, including, *Direct Methods for Sparse Matrices* by Iain Duff (1987), *Direct Methods for Sparse Linear Systems*, by Tim Davis (2006), etc.

How can we improve the complexity?

Let us return to our basic model problem:



Recall that the coefficient matrix is tessellated as

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

First consider the case of the standard 5-point stencil:

$$[\mathbf{Au}](k) = \frac{1}{h^2}\left(4\,\mathbf{u}(k) - \mathbf{u}(k_\mathrm{s}) - \mathbf{u}(k_\mathrm{e}) - \mathbf{u}(k_\mathrm{n}) - \mathbf{u}(k_\mathrm{w})\right)$$

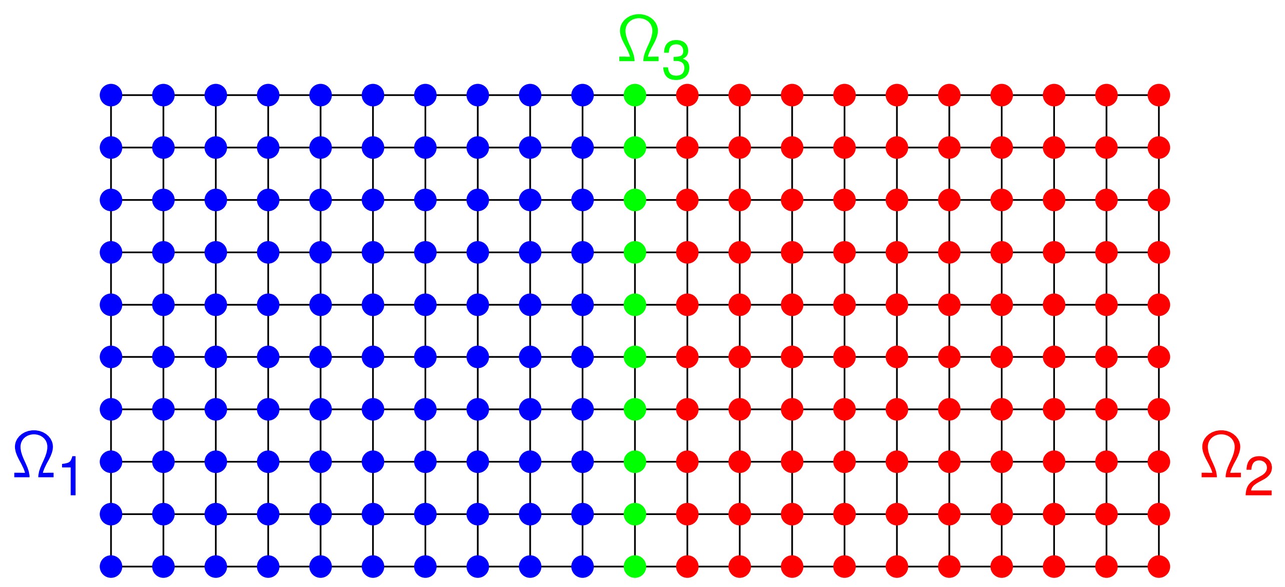
Recall the definition of the Schur complement: $\mathbf{S}_3 = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$

The cost of an LU-factorization is dominated by the cost of factoring the dense Schur complement of size $\sqrt{N} \times \sqrt{N}$ at the top level.

Question: Can this step be accelerated?

How can we improve the complexity?

Let us return to our basic model problem:



Recall that the coefficient matrix is tessellated as

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

First consider the case of the standard 5-point stencil:

$$[\mathbf{Au}](k) = \frac{1}{h^2}\left(4\,\mathbf{u}(k) - \mathbf{u}(k_{\rm s}) - \mathbf{u}(k_{\rm e}) - \mathbf{u}(k_{\rm n}) - \mathbf{u}(k_{\rm w})\right)$$

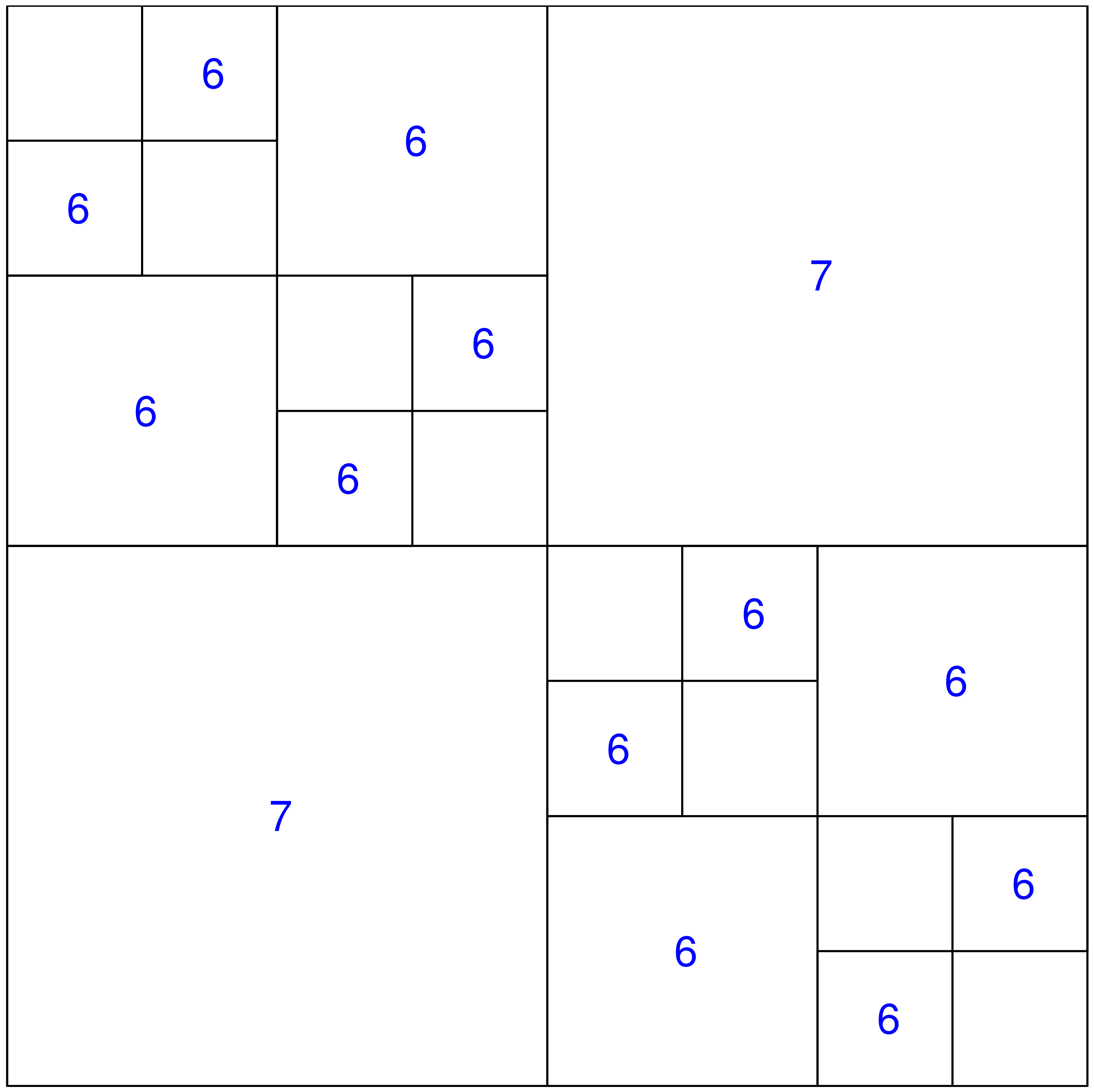
Recall the definition of the Schur complement: $\mathbf{S}_3 = \mathbf{A}_{33} - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}$

The cost of an LU-factorization is dominated by the cost of factoring the dense Schur complement of size $\sqrt{N} \times \sqrt{N}$ at the top level.

Question: Can this step be accelerated?

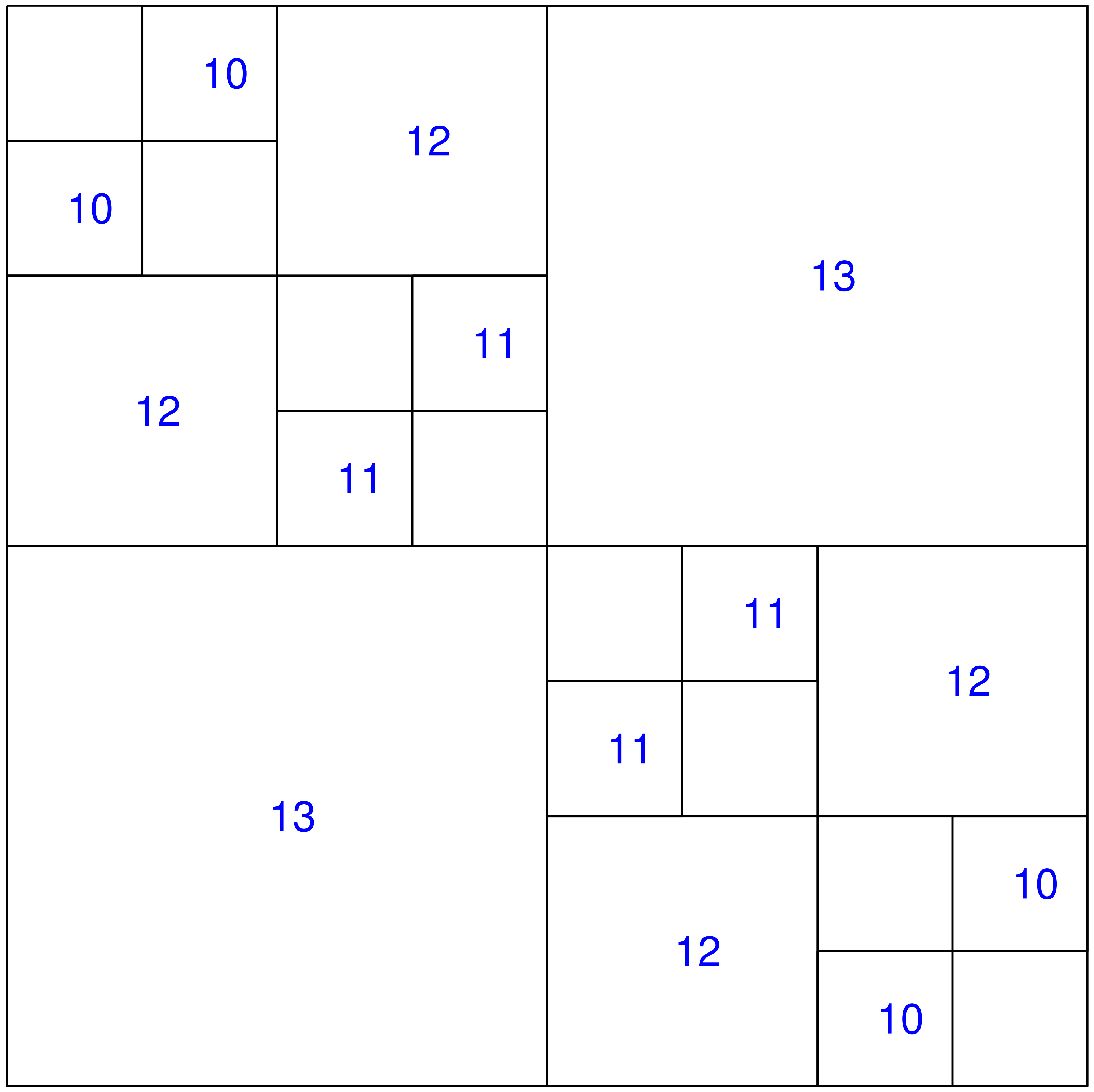
Answer: Yes, it turns out that $\mathbf{S}_3$ is often *rank-structured.*

Ranks structure of S_3. acc=1.00e−05 nside=400 type=orig

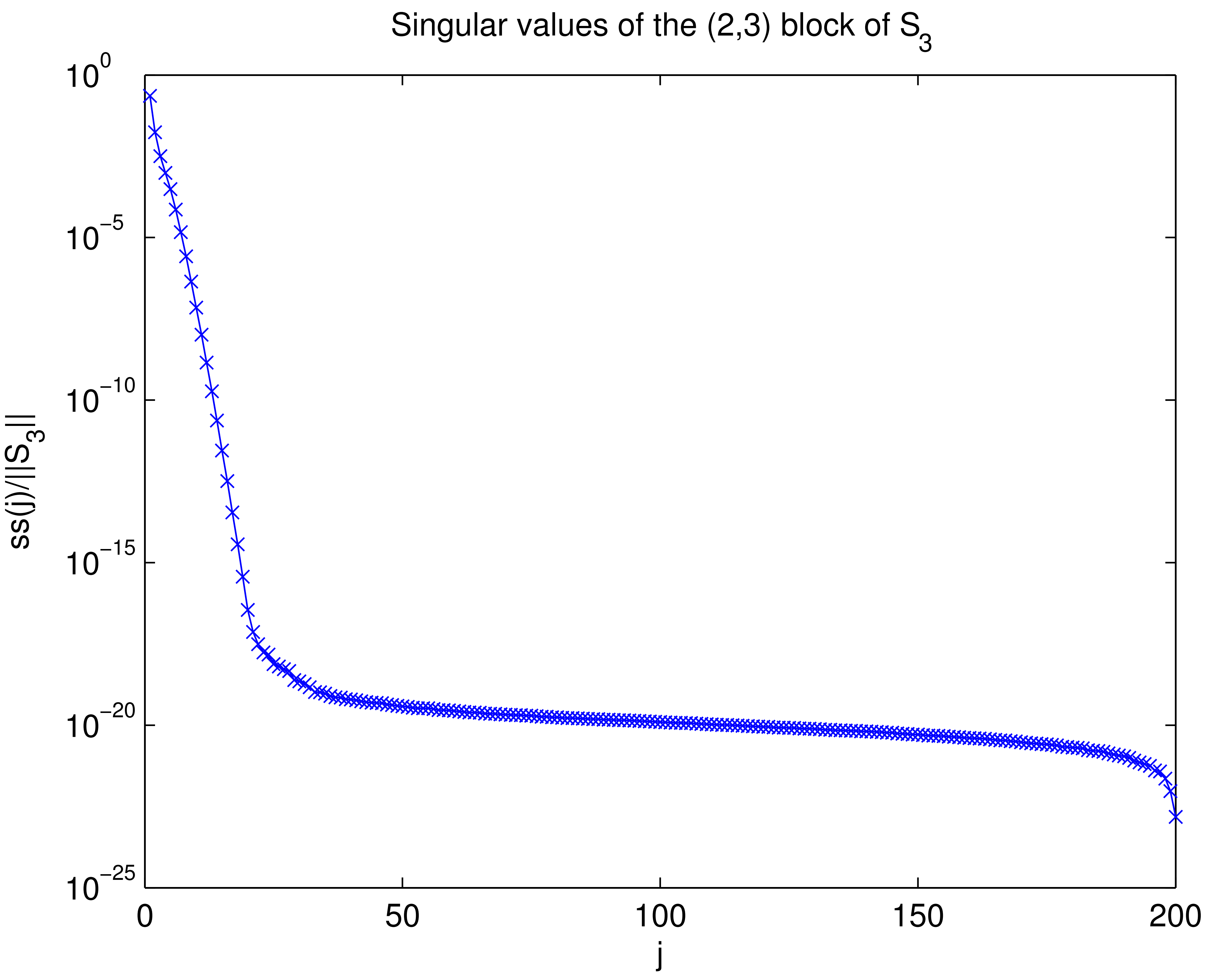


Plain 5-point stencil. Relative accuracy = 10^{-5}.

Ranks structure of S_3. acc=1.00e−10 nside=400   type=orig

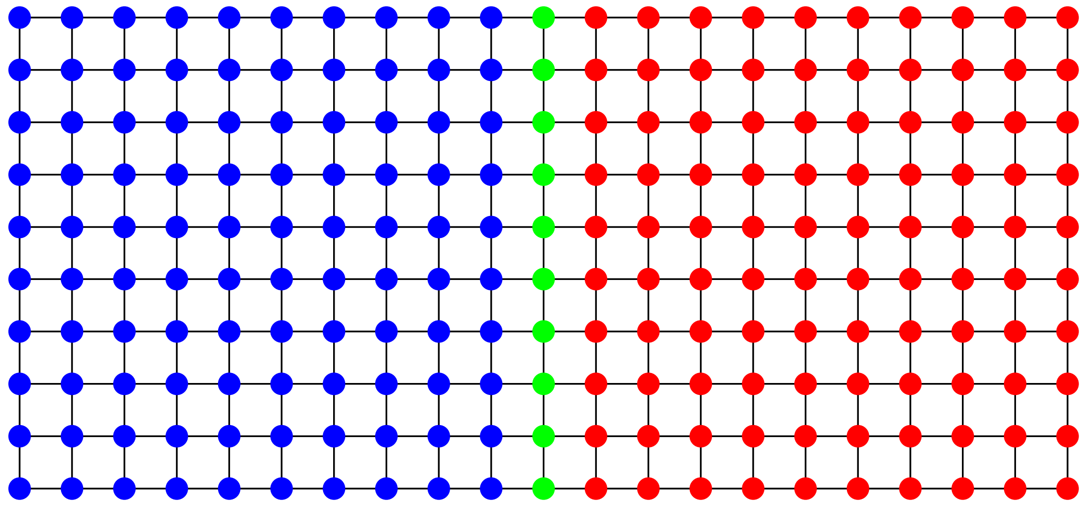


Plain 5-point stencil. Relative accuracy = 10^{-10}.



Singular values of the top-right quadrant of $\mathbf{S}_3$ (plain 5-point stencil).

Maybe the reason for the extreme decay is some peculiarity of the plain 5-point stencil. Possibly that it has very smooth (constant!) coefficients?



Next, let us try a *grid conduction problem*:

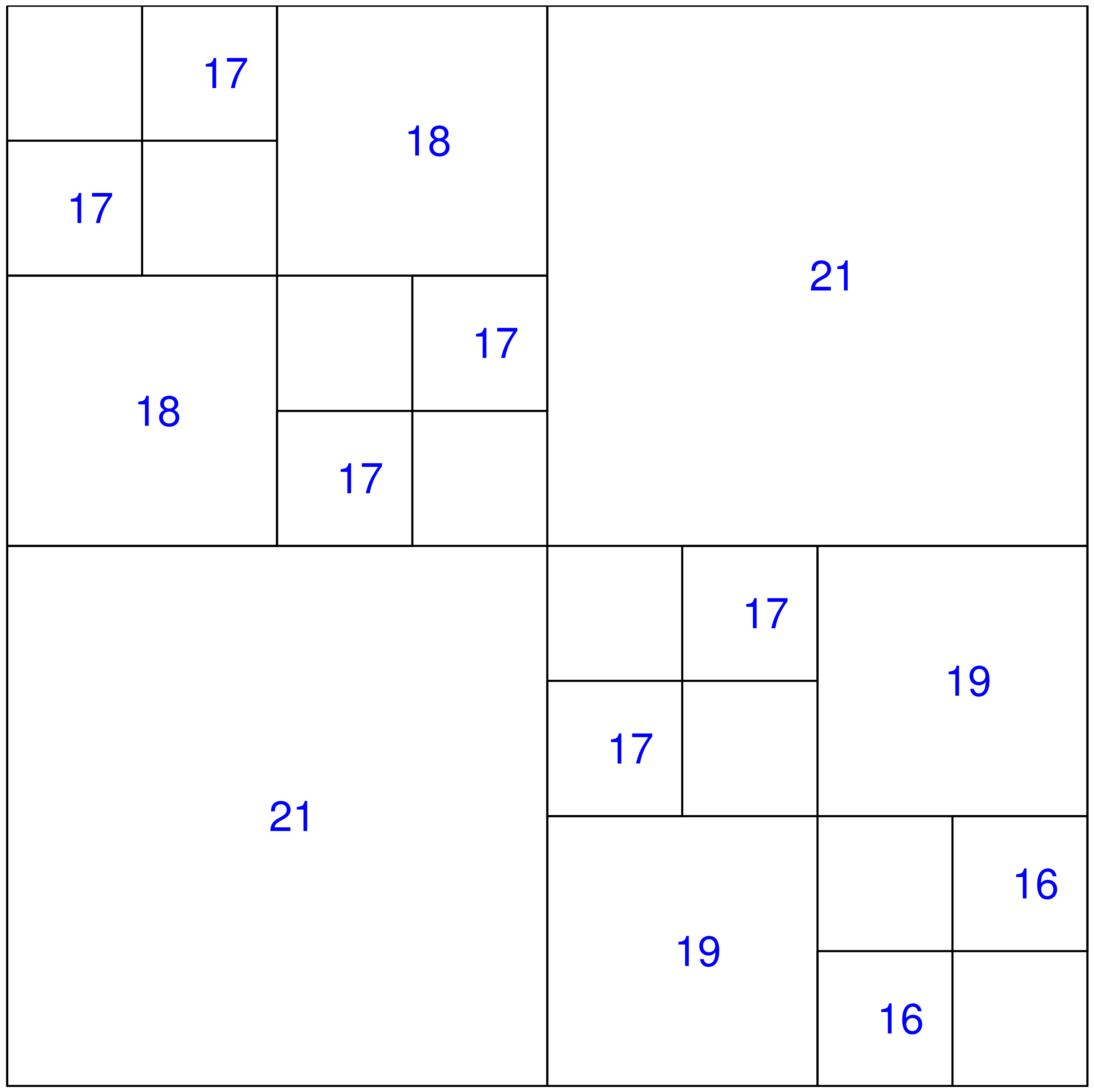
$$[\mathbf{Au}](k) = \alpha_{k,k_\mathrm{e}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{e})\big)+\alpha_{k,k_\mathrm{n}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{n})\big)+\alpha_{k,k_\mathrm{w}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{w})\big)+\alpha_{k,k_\mathrm{s}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{s})\big),$$

where $\alpha_{k,\ell}$ is the conductivity of the link connecting nodes k and ℓ.

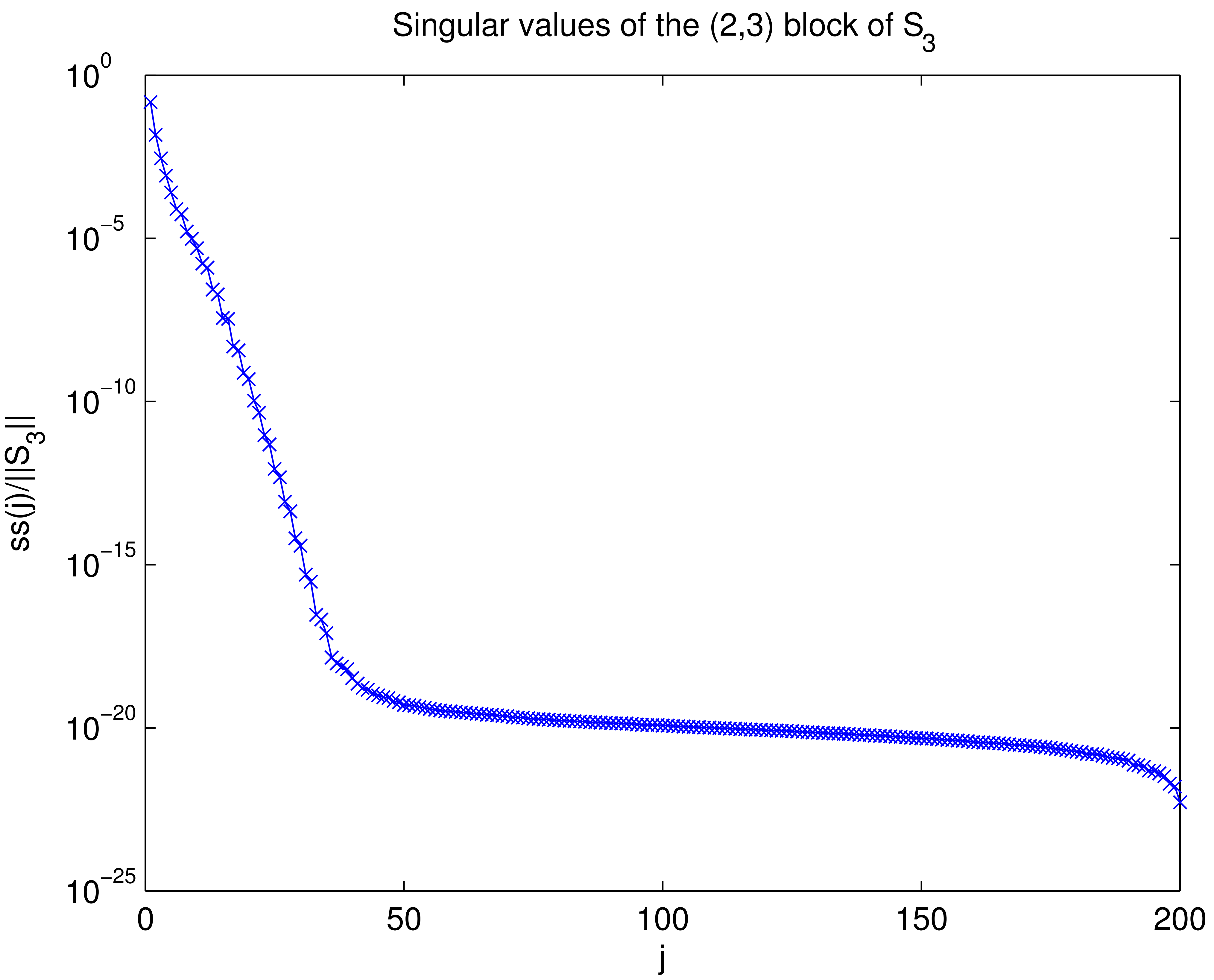
This is a purely discrete problem, there is really no “underlying” PDE here.

First, we’ll draw the conductivities from a uniform distribution on $[1,2]$.

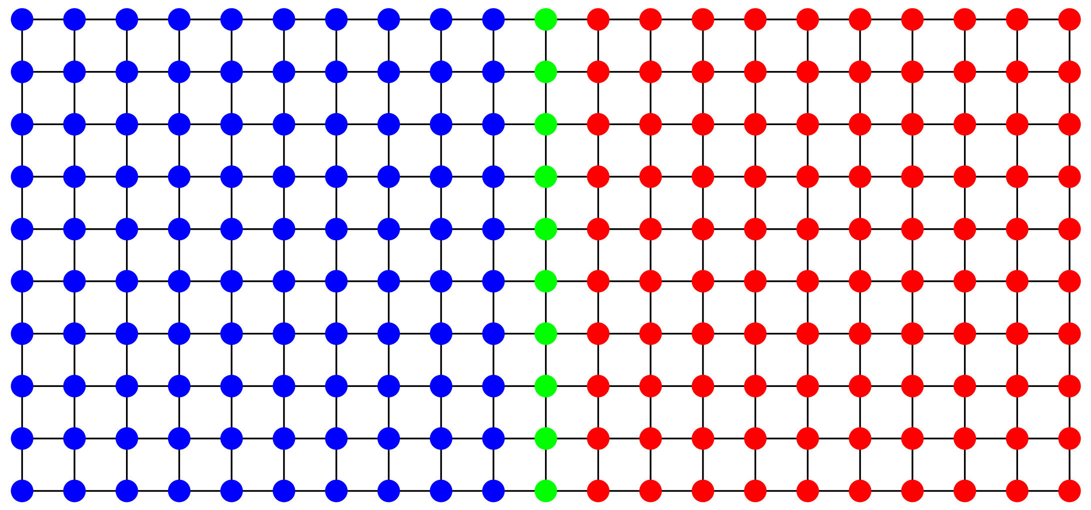
Ranks structure of S_3. acc=1.00e−10 nside=400 type=rcon



Random conductivity network, $\alpha_{k,j} \in U([1,2])$. *Relative accuracy* = 10^{-10}.



Singular values of the top-right quadrant of $\mathbf{S}_3$ *(random network,* $\alpha_{k,j} \in U([1,2])$*).*



We again do a grid conduction problem:

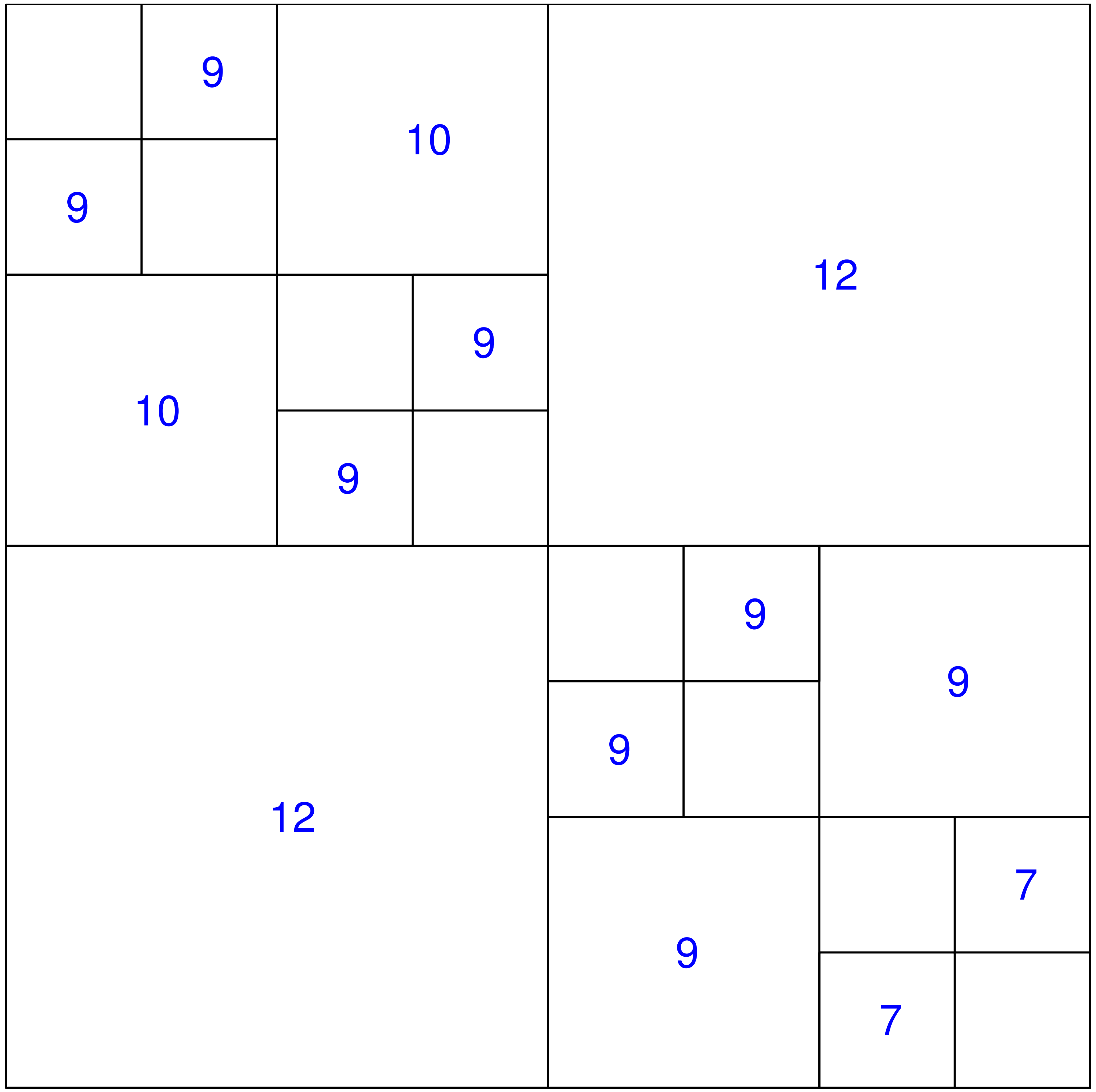
$$[\mathbf{Au}](k) = \alpha_{k,k_\mathrm{e}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{e})\big)+\alpha_{k,k_\mathrm{n}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{n})\big)+\alpha_{k,k_\mathrm{w}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{w})\big)+\alpha_{k,k_\mathrm{s}}\big(\mathbf{u}(k)-\mathbf{u}(k_\mathrm{s})\big),$$

where $\alpha_{k,\ell}$ is the conductivity of the link connecting nodes k and ℓ.

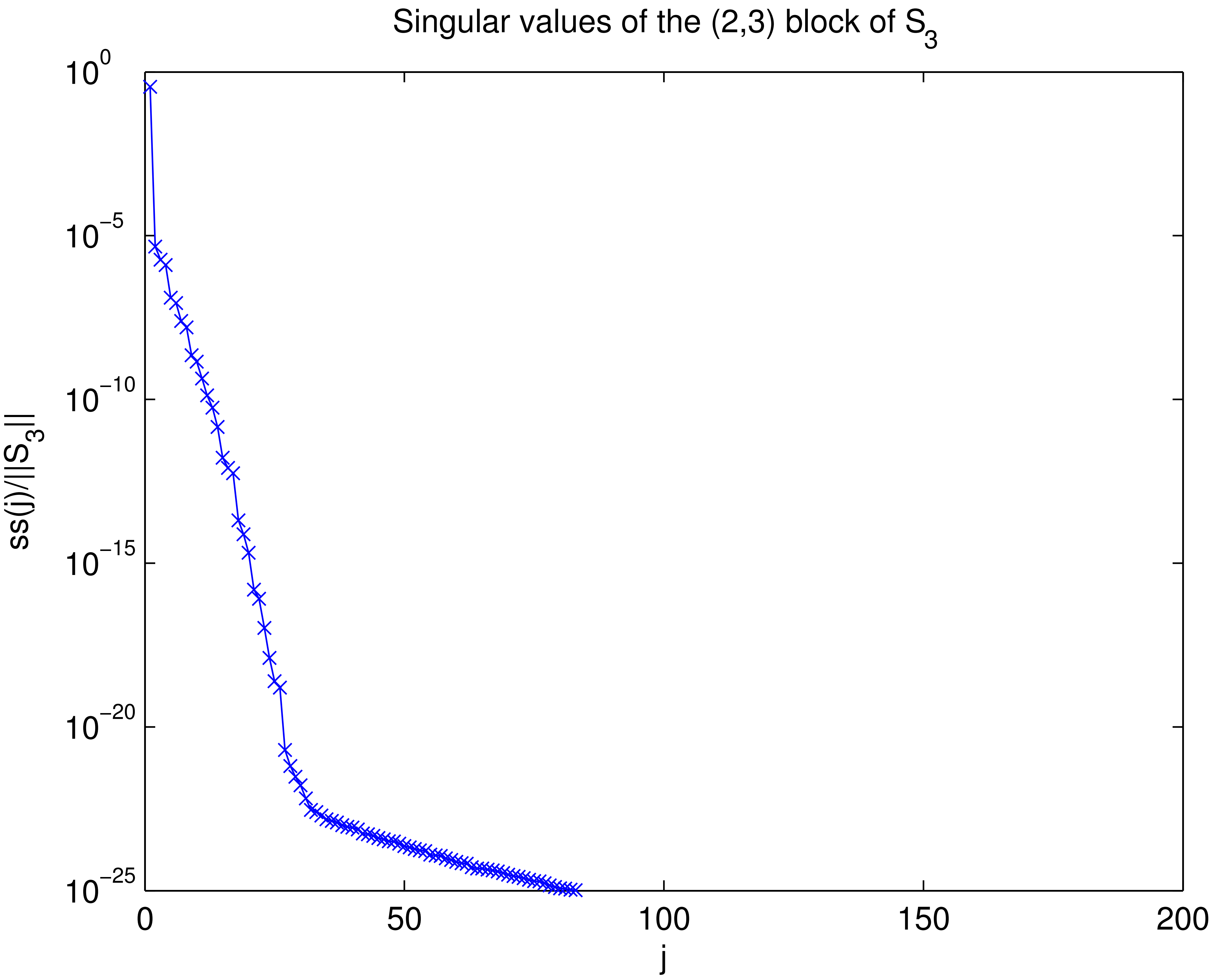
Now, let us crank up the aspect ration, set $\alpha_{k,j} = 10^{-\beta_{k,j}}$ where $\beta_{k,j} \in U([0, 10])$

In other words, the conductivities will be drawn at random in the interval $[10^{-10}, 1]$.

Ranks structure of S_3. acc=1.00e−10 nside=400 type=rcom

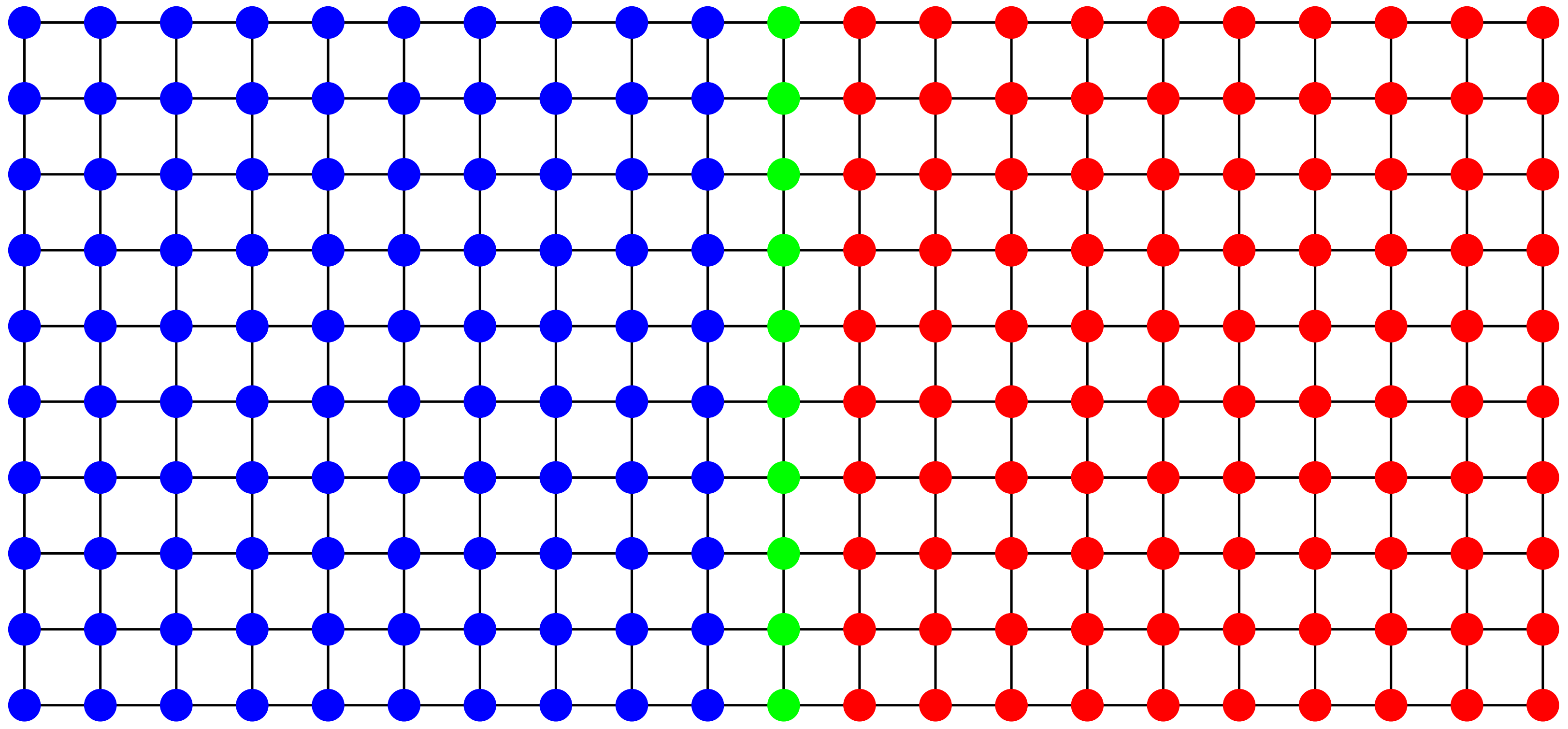


Random conductivity network, high aspect ratio. Relative accuracy = 10^{-10}.



Singular values of the top-right quadrant of $\mathbf{S}_3$ *(high aspect ratio random network).*

Next, let us try a *Helmholtz problem.*



Define the discrete Laplace operator via

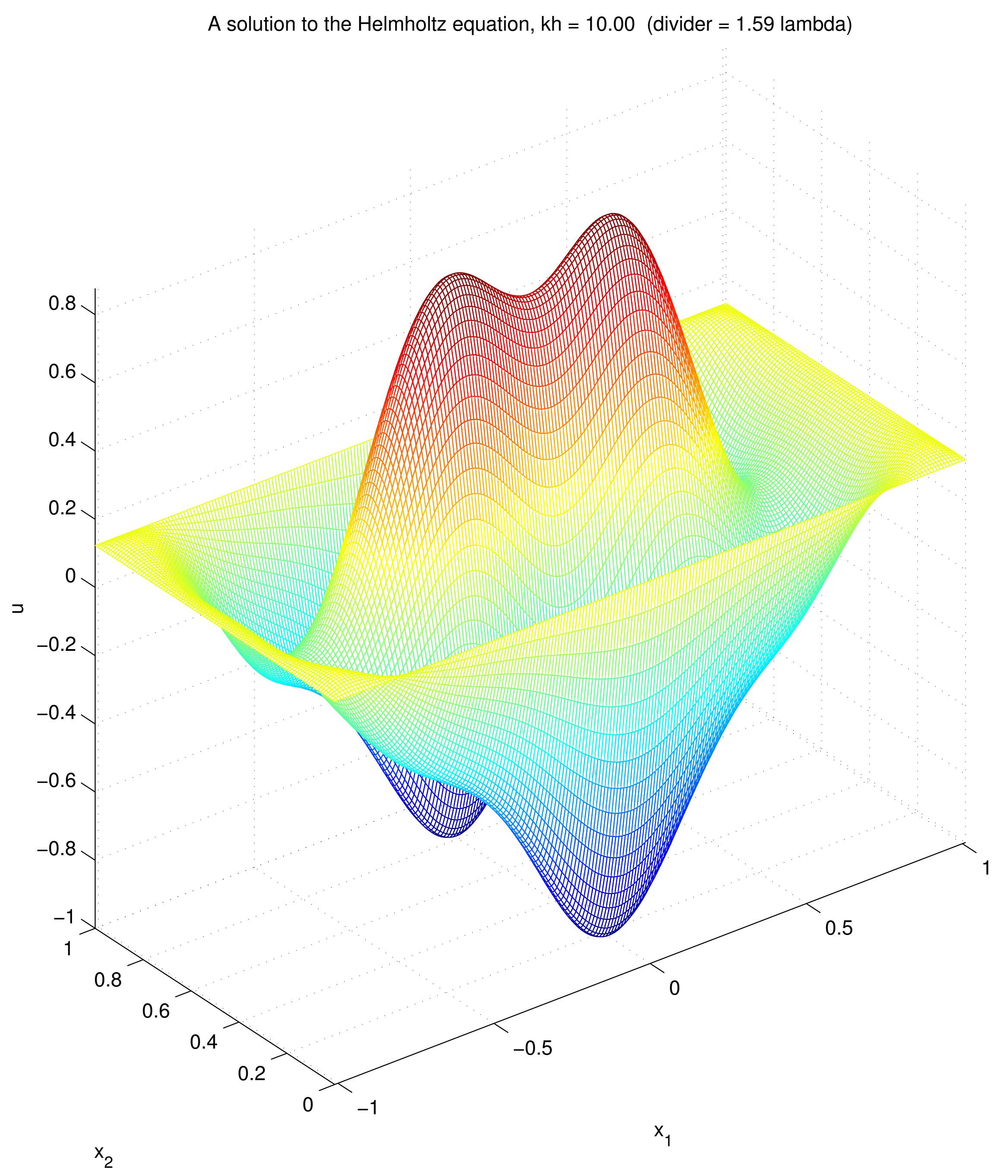
$$[\mathbf{L}\mathbf{u}](k) = \frac{1}{h^2}\left(4\,\mathbf{u}(k) - \mathbf{u}(k_{\rm s}) - \mathbf{u}(k_{\rm e}) - \mathbf{u}(k_{\rm n}) - \mathbf{u}(k_{\rm w})\right)$$

and then the Helmholtz operator via

$$\mathbf{A} = \mathbf{L} - \kappa^2\mathbf{I},$$

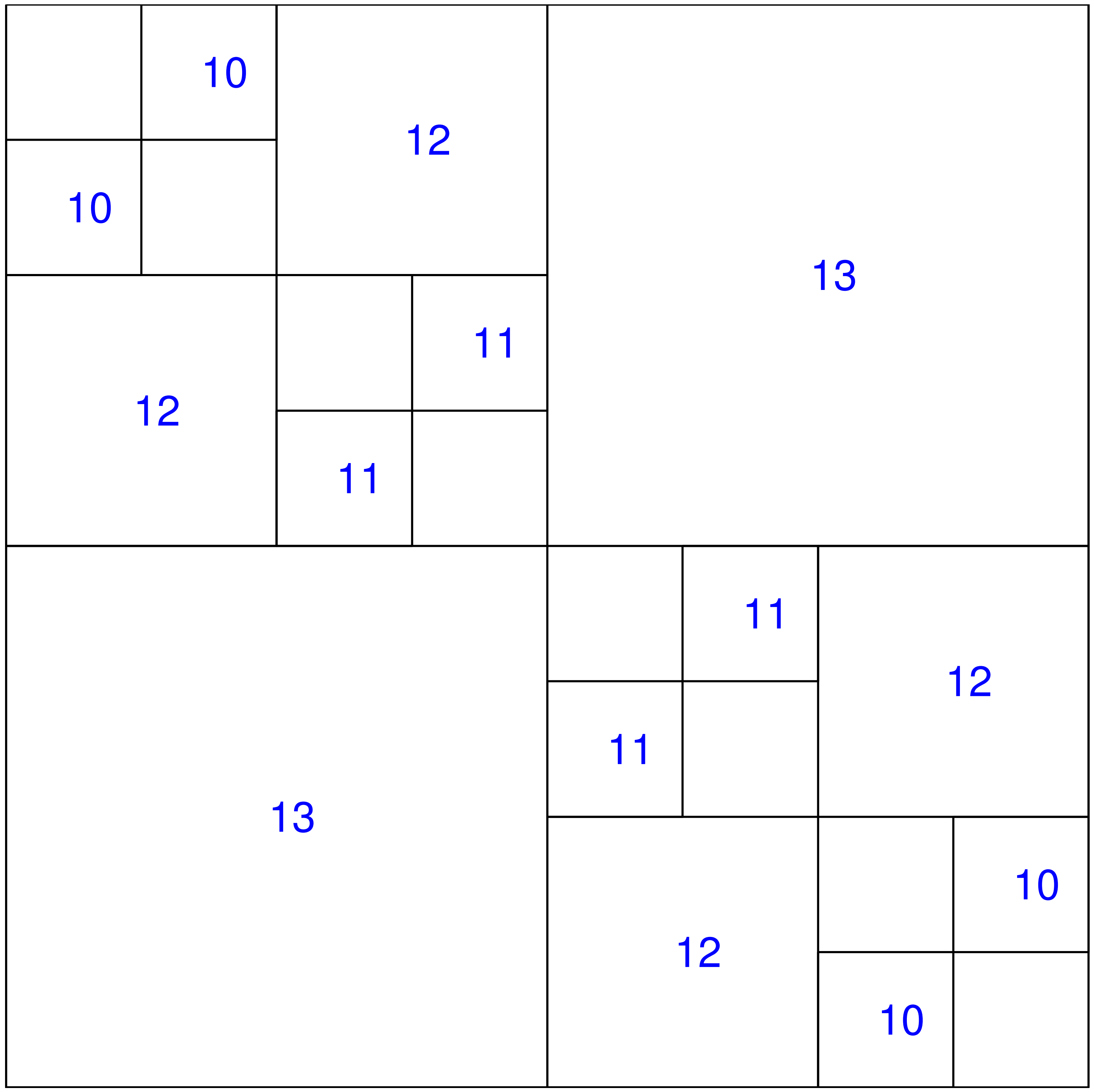
where κ is the wave-number.

A representative solution to the Helmholtz problem with $\kappa = 10$, (divider is 1.6λ high).

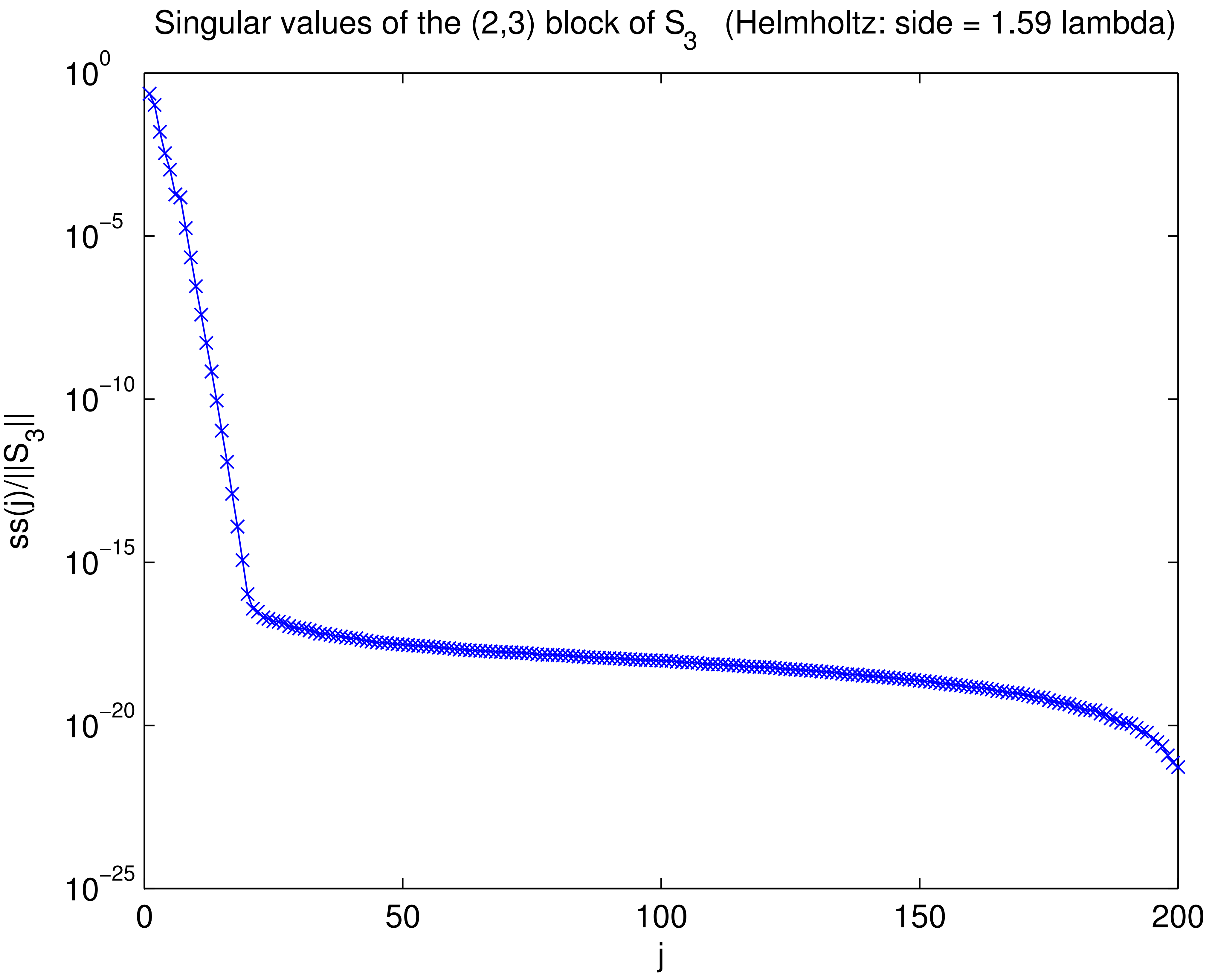


Ranks for Helmholtz, divider is 1.6λ*, accuracy* = 10^{-10}.

Rank structure of S_3. acc=1.00e−10 nside=400 (Helmholtz: side = 1.59 lambda)

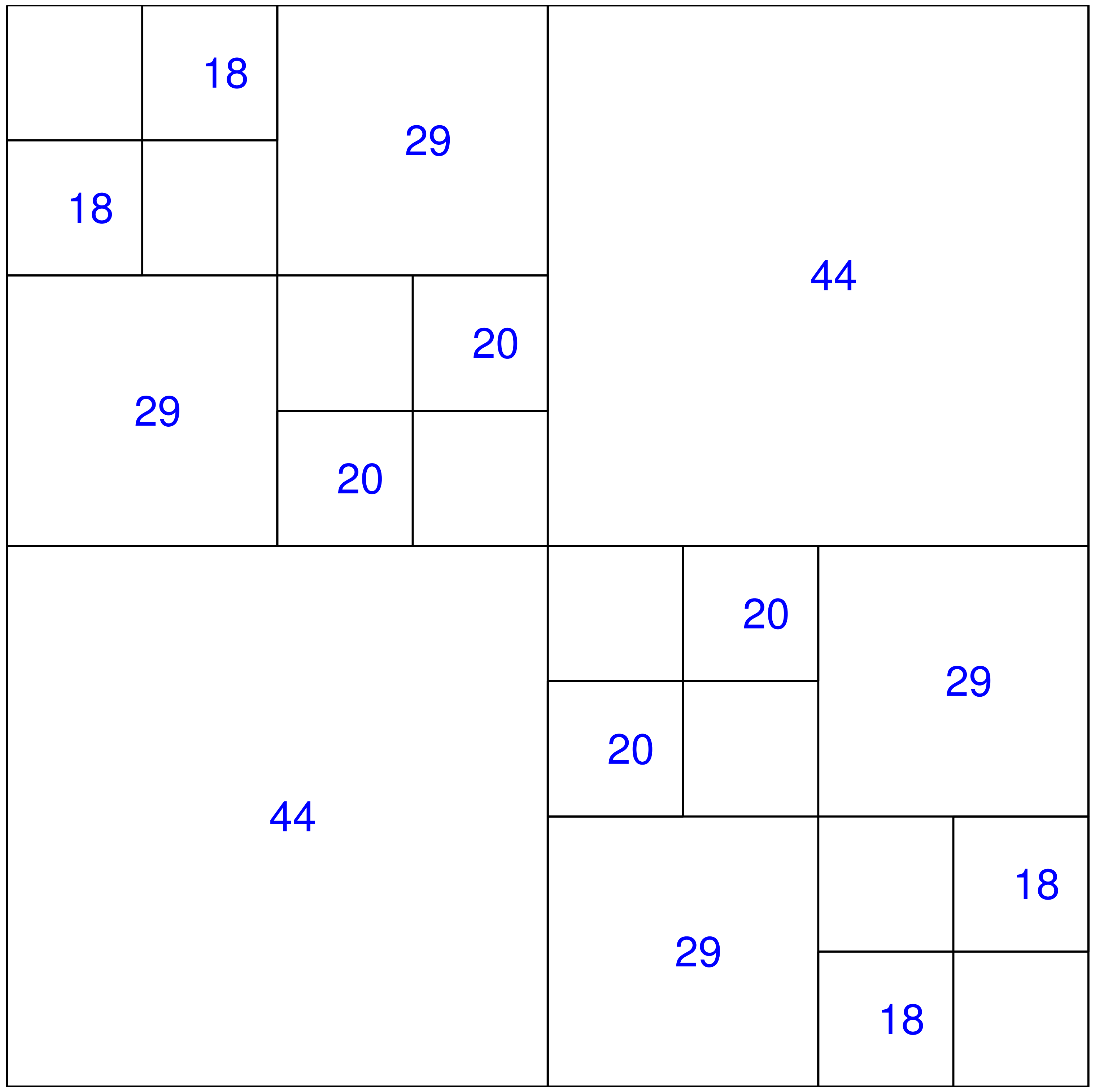


Singular values for Helmholtz, divider is 1.6λ.

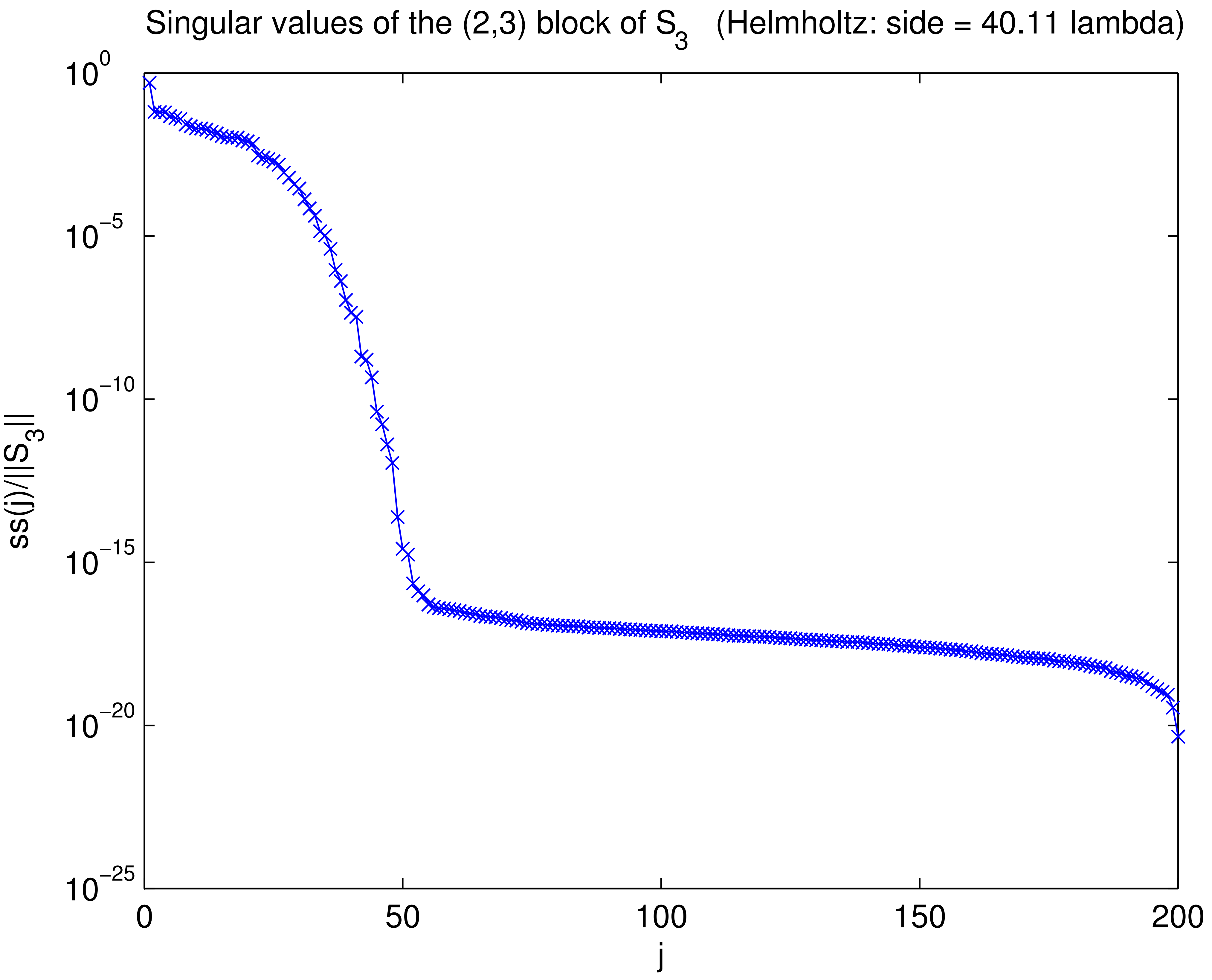


Ranks for Helmholtz, divider is 40λ, accuracy = 10^{-10}.

Rank structure of S_3. acc=1.00e−10 nside=400 (Helmholtz: side = 40.11 lambda)

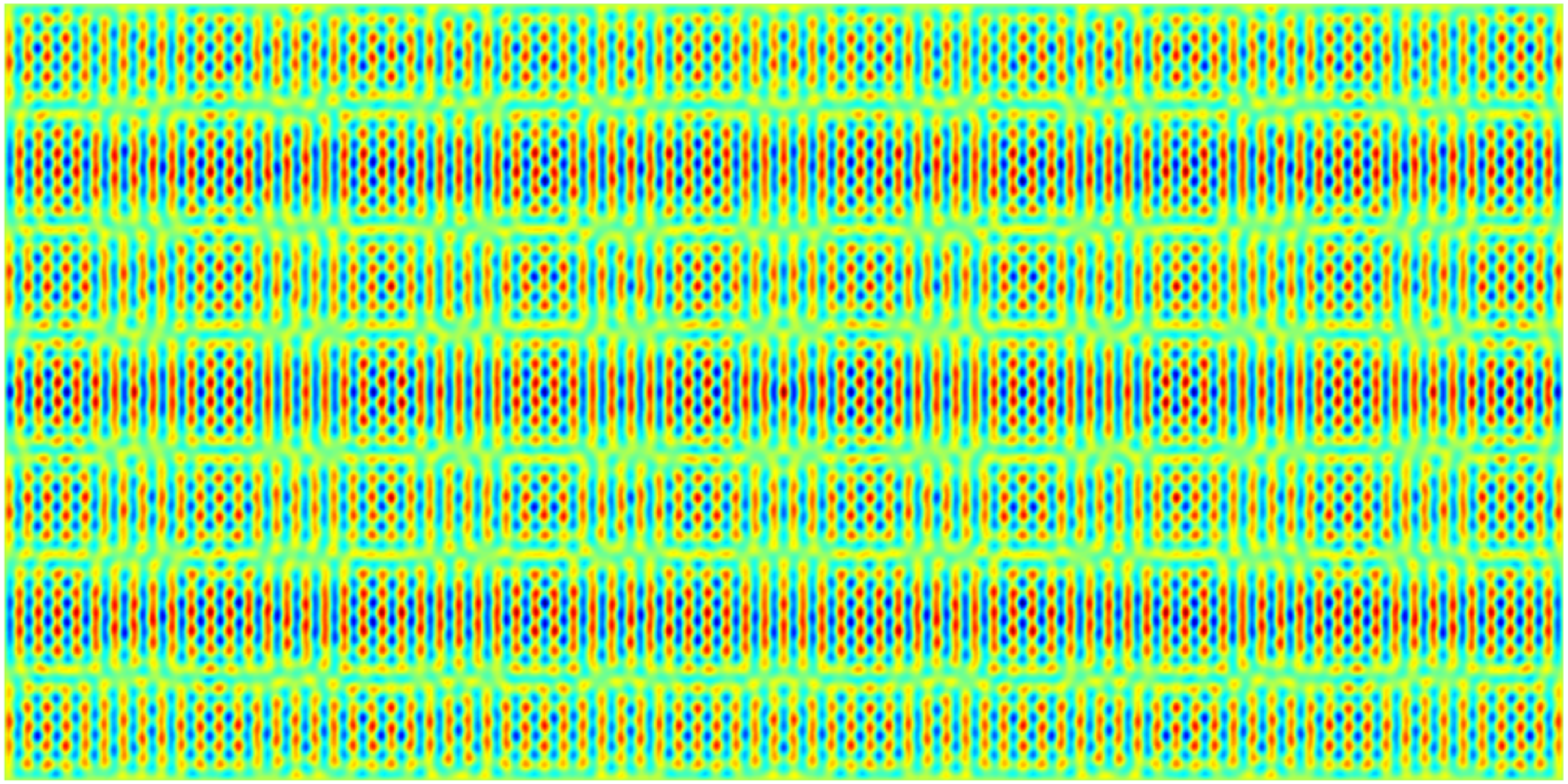


Singular values for Helmholtz, divider is 40λ.



A representative solution to the Helmholtz problem with $\kappa = 252$, (divider is 40λ high).

A solution to the Helmholtz equation, kh = 252.00 (divider = 40.11 lambda)



Factors that influence the rank structure

Disclaimer: This topic is not well understood ...

- For Laplace-type problems, the ranks are consistently small.
- Smoothness of coefficients do not seem to matter at all.
  The exponential decay is preserved for “diffusive networks” with no PDE analog.
- Fairly insensitive to precise form of operator (e.g. convection diffusion).
- For problems with oscillatory solutions, the rank increases as the wave-length shrinks.
- Some theory exists, but it is quite weak $\rightarrow$ opportunity for more work!

So how fast does the algebra for handing Schur complements have to be?

Suppose that the cost of inverting (or factoring) an $m \times m$ Schur complement $\mathbf{S}$ is $\sim m^{\alpha}$.

Then the asymptotic costs of nested dissection get reduced as follows:

Cost of processing level 0:	$\sim (N^{1/2})^{\alpha}$	$\sim N^{\alpha/2}$
Cost of processing level 1:	$\sim 2\,((N/2)^{1/2})^{\alpha}$	$\sim 2^{1-\alpha/2} N^{\alpha/2}$
Cost of processing level 2:	$\sim 4\,((N/4)^{1/2})^{\alpha}$	$\sim 4^{1-\alpha/2} N^{\alpha/2}$
Cost of processing level 3:	$\sim 8\,((N/8)^{1/2})^{\alpha}$	$\sim 8^{1-\alpha/2} N^{\alpha/2}$
$\vdots$	$\vdots$	$\vdots$

If $\alpha > 2$, then the cost is $\sim N^{\alpha/2}$.

If $\alpha = 2$, then the cost is $\sim N \log N$.

If $\alpha < 2$, then the cost is $\sim N$.

For problems in 2D, we can easily manipulate $\mathbf{S}_{\tau}$ in $\sim N\,(\log N)^2$ operations.

So how fast does the algebra for handing Schur complements have to be? — 3D

Suppose that the cost of inverting (or factoring) an $m \times m$ Schur complement $\mathbf{S}$ is $\sim m^{\alpha}$.

Then the asymptotic costs of nested dissection get reduced as follows:

Cost of processing level 0:	$\sim (N^{2/3})^{\alpha}$	$\sim N^{2\alpha/3}$
Cost of processing level 1:	$\sim 2\,((N/2)^{2/3})^{\alpha}$	$\sim 2^{1-2\alpha/3} N^{2\alpha/3}$
Cost of processing level 2:	$\sim 4\,((N/4)^{2/3})^{\alpha}$	$\sim 4^{1-2\alpha/3} N^{2\alpha/3}$
Cost of processing level 3:	$\sim 8\,((N/8)^{2/3})^{\alpha}$	$\sim 8^{1-2\alpha/3} N^{2\alpha/3}$
$\vdots$	$\vdots$	$\vdots$

If $\alpha > 3/2$, then the cost is $\sim N^{2\alpha/3}$.

If $\alpha = 3/2$, then the cost is $\sim N \log N$.

If $\alpha < 3/2$, then the cost is $\sim N$.

Summary

1. *Construct a quad-tree:* Partition the grid into a hierarchy of boxes.
2. *Process the leaves:* For each leaf box in the tree, construct its Schur complement.
3. *Hierarchical merge:* Loop over all levels of the tree, from finer to smaller. For each box on a level, compute its Schur complement by merging the (already computed) Schur complements of its children.
4. *Process the root of the tree:* After completing Step 3, the Schur complement for the entire domain is available. Invert (or factor) it to construct the solution operator.

Remark: For simplicity, the algorithm is described in a level-by-level manner (process all leaves first, then proceed one level at a time in going upwards). In fact, there is flexibility to travel through the tree in any order that ensures that no node is processed before its children. Since all Schur complements can be discarded once their information has been passed on to a parent, smarter orderings can greatly reduce the memory requirements.

Bibliography: *See website!* L. Grasedyck, S. Le Borne, and R. Kriemann (2007); Martinsson (2009); J. Xia, S. Chandrasekaran, M. Gu, and X.S. Li (2009) and many following papers by Xia et al; Gillman (2011); Schmitz & Ying (2012); Gillman & Martinsson (2014), Ho & Ying (2014); current work by Eric Darve's group, . . .