Hierarchical Solvers

Eric Darve

CM4 Summer School

June 2016

Matrices and linear solvers

- How can we solve Ax = b?
- Direct methods: Gaussian elimination, LU and QR factorizations: $O(n^3)$
- Iterative methods: GMRES, Conjugate Gradient, MINRES, etc

Iterative Methods

- Iterative methods can be very fast.
- They rely primarily on matrix-vector products Ax.
- If A is sparse this can be done very quickly.
- However, the convergence of iterative methods depends on the distribution of eigenvalues.
- So it may be quite slow in many instances.

Conjugate Gradient

- In the case of conjugate gradient, the convergence analysis is quite simplified.
- The key result is as follows:

Error at step n

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq \inf_{p \in P_n} \max_{\lambda \in \Lambda(A)} |p(\lambda)|$$

$p \in P_n$: polynomials of degree less than n with $p(0) = 1$

$\Lambda(A)$ is the set of all eigenvalues of A.

Canonical cases

- If all the eigenvalues are clustered around a few points (say around 1), then convergence is fast.
- Just place all the roots of p inside each cluster of eigenvalues.

Ill-conditioned case

- Recall that p(0) = 1.
- So if some eigenvalues are very close to 0, while others are far away, it is difficult to minimize p(λ).
- For CG:

$$\frac{\|e_n\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \sim 2\,(1-2/\sqrt{k})^n$$

Difficulty when condition number κ is large

Preconditioners

- Most engineering matrices are not well-conditioned and have eigenvalues that are not well distributed.
- To solve such systems, a preconditioner is required.
- The effect of the preconditioner will be to regroup the eigenvalues into a few clusters.

Hierarchical solvers

- Hierarchical solvers offer a bridge between direct and iteration solvers.
- They lead to efficient preconditioners suitable for iterative techniques.
- They are based on **approximate direct factorizations of the matrix.**
- **Computational cost is O(n) for many applications (depending on properties of matrix).**

Cost of factorization

- The problem with direct methods and matrix factorization is that they lead to a large computational cost.
- Matrix of size n: cost is $O(n^3)$.
- This problem can be mitigated for sparse matrices with many zeros.
- **Hierarchical solvers offer a trade-off between computational cost and accuracy for direct methods.**

Factorization for sparse matrices

Assume we start from a sparse matrix and perform one step of a block LU factorization:

$$A = \begin{pmatrix} I & \\ UA_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & \\ 0 & A_{22} - UA_{11}^{-1}U^T \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1}U^T \\ & I \end{pmatrix}$$

This block may have a lot of new non-zero entries

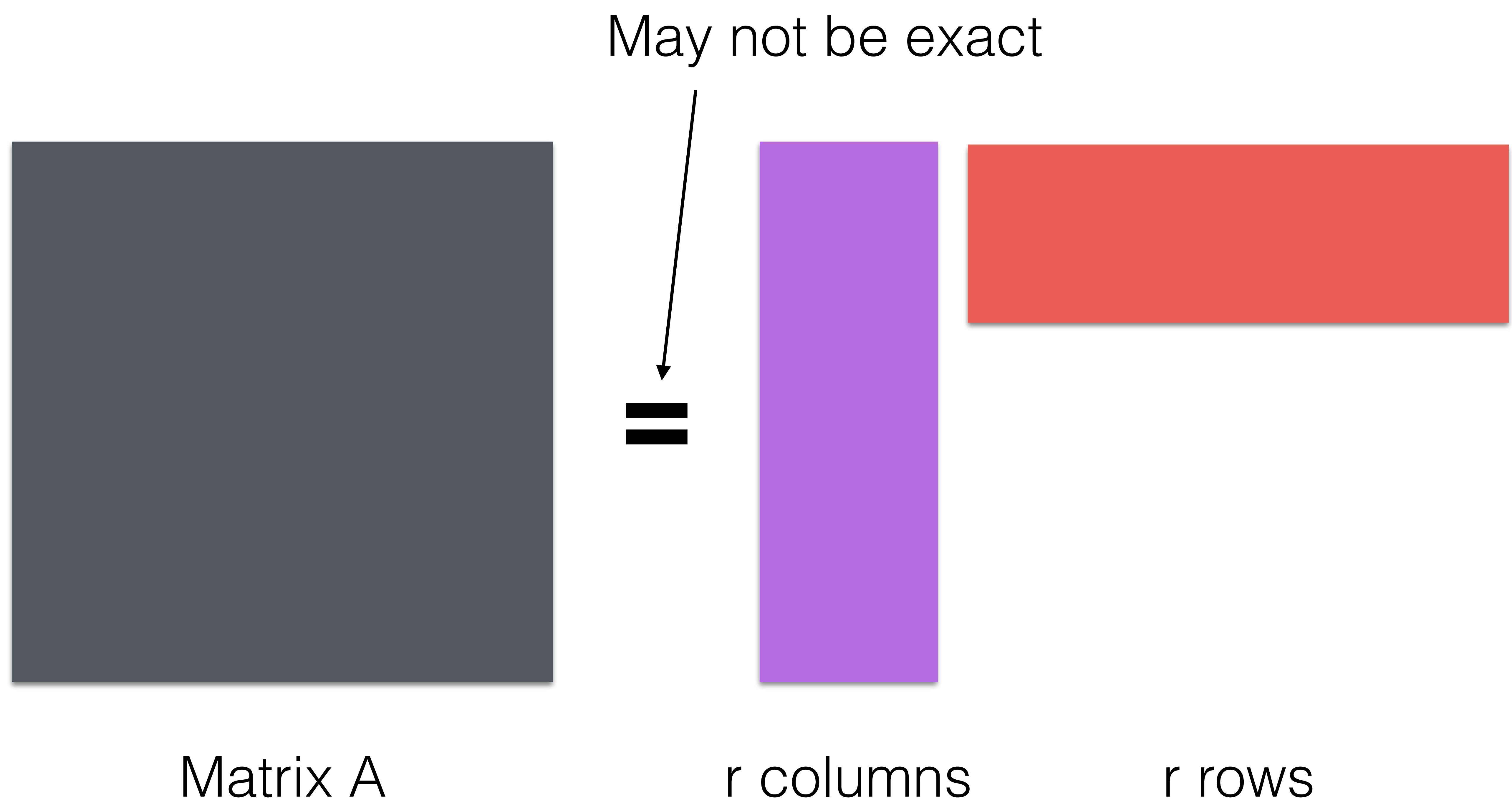
Sparsification

- Hierarchical methods attempt to maintain the sparsity of the matrix to prevent the fill-in we just discovered.
- How does it work?

Low-rank

- The basic mechanism is to take advantage of the fact that dense blocks can often be approximated by a low-rank matrix.
- This is not always true though. We will investigate this in more details during the tutorial session.
- Canonical case: for elliptic PDEs, this low-rank property is always observed for clusters of points in the mesh that are well-separated (à la fast multipole method).

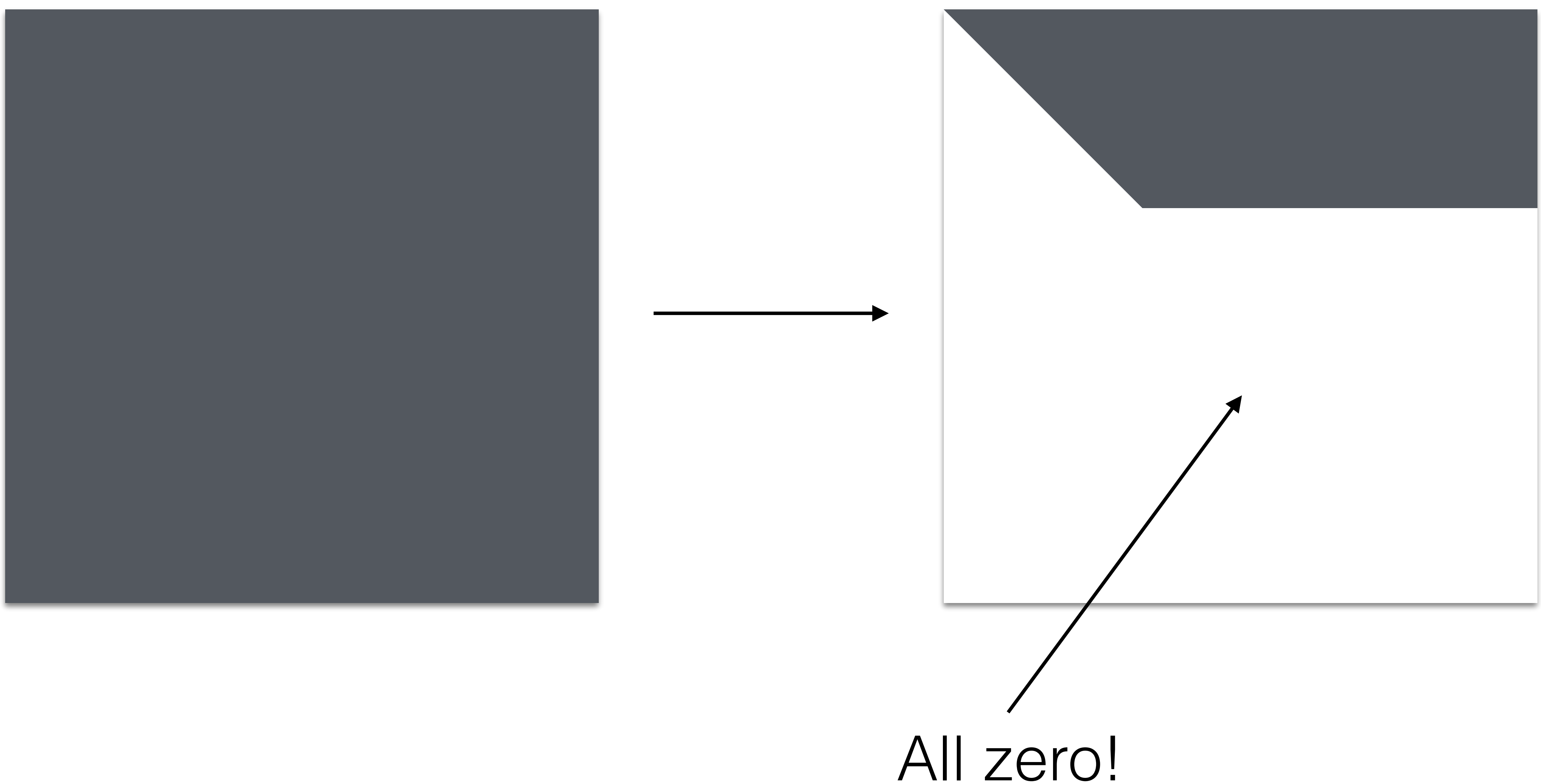
What is a low-rank matrix?



LU

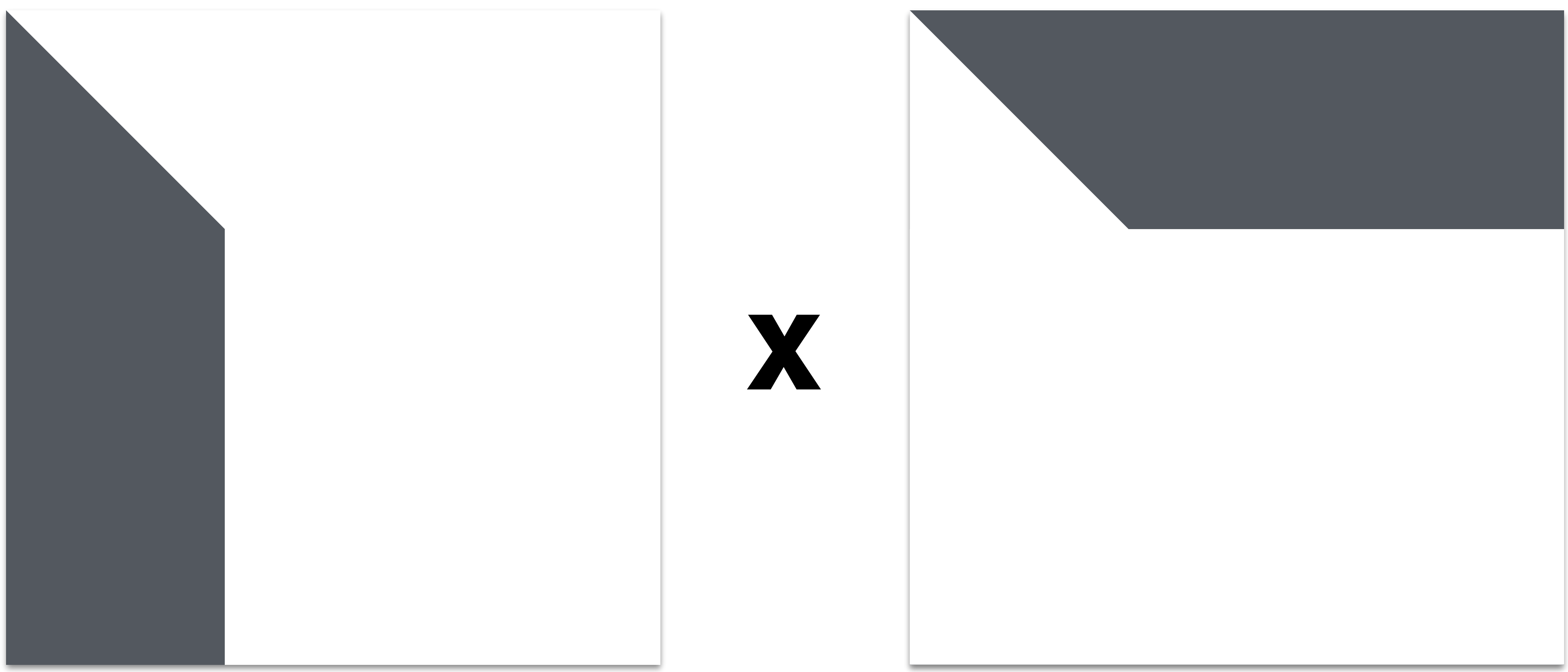
- LU factorizations are a great tool for low-rank matrices.
- Assume we have a low-rank matrix and we perform an LU factorization (with full pivoting).
- What happens?

LU for low-rank



Low-rank factorization

In fact, LU directly produces a factorization of the form:



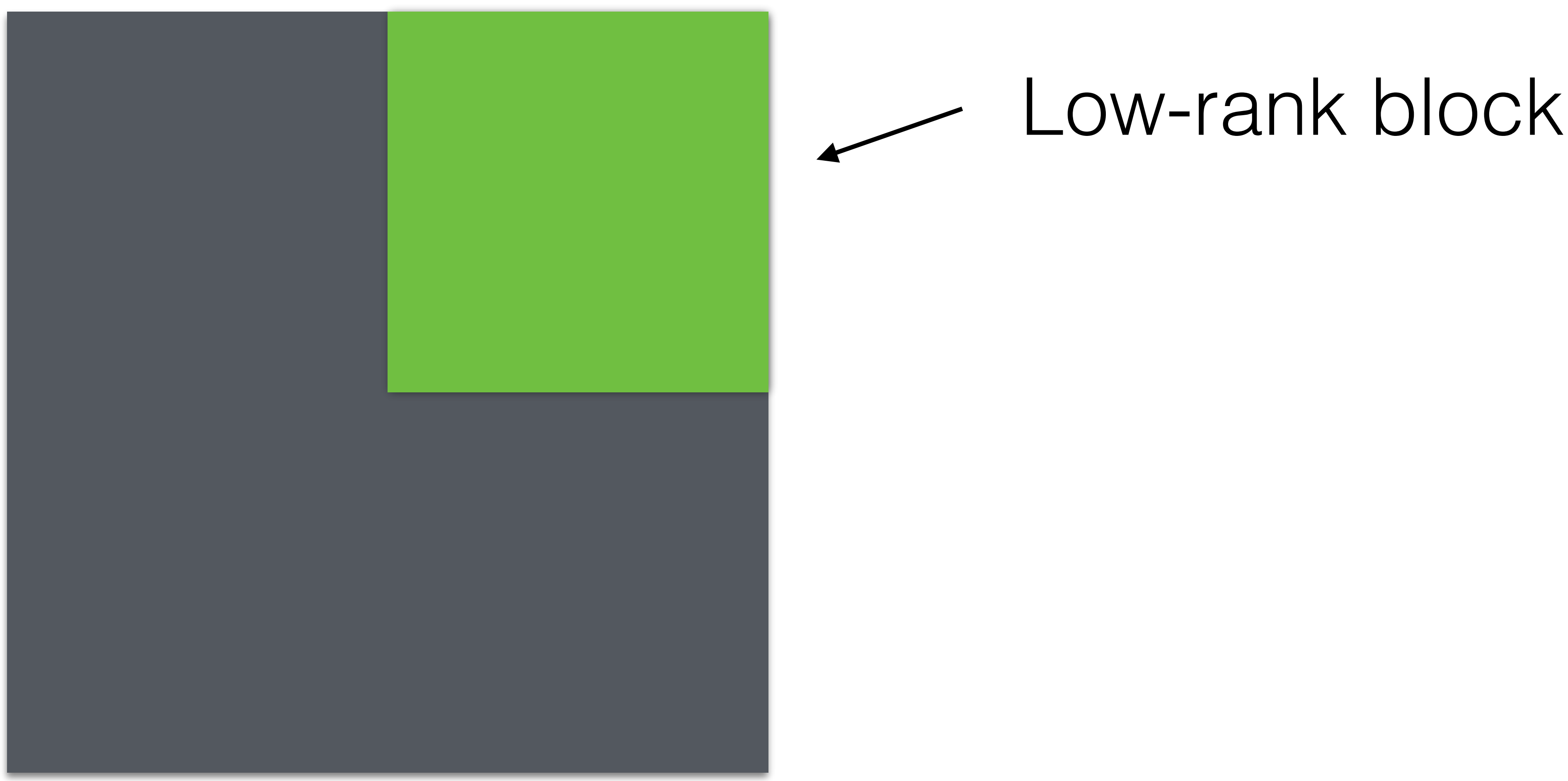
L factor

U factor

How can we use this?

- Let’s see how we can apply this to remove entries in our matrix.
- Recall that the factorization leads to a lot of fill-in.
- LU comes to the rescue to restore sparsity!

Matrix with low-rank block



Create a new block of 0

Sparsity

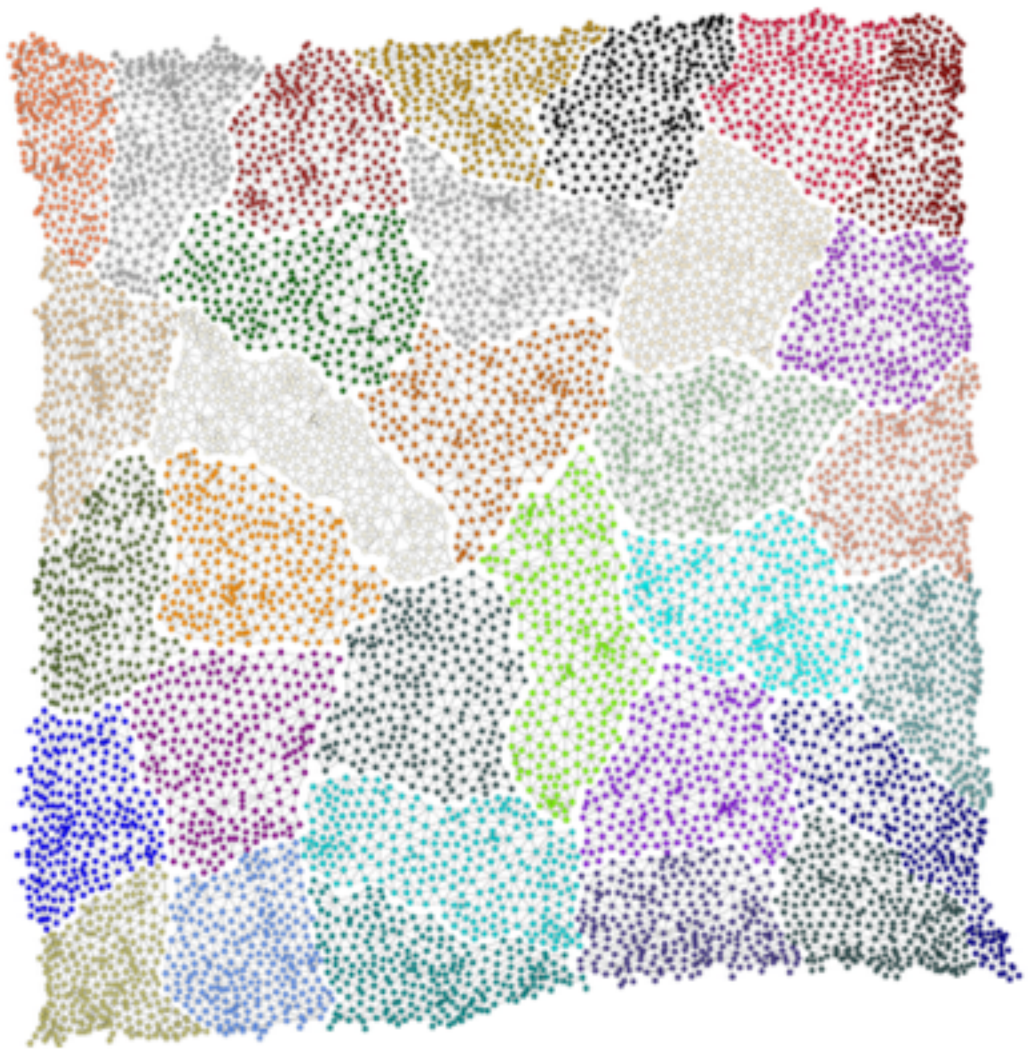
The fast factorization scheme proceeds as follows:

- Perform a Cholesky or LU factorization.
- When a new fill-in occurs in a block corresponding to well-separated nodes (say in the mesh for a discretized PDE), use row transformations to sparsify the matrix.

This process allows factoring A into a product of completely sparse matrices!

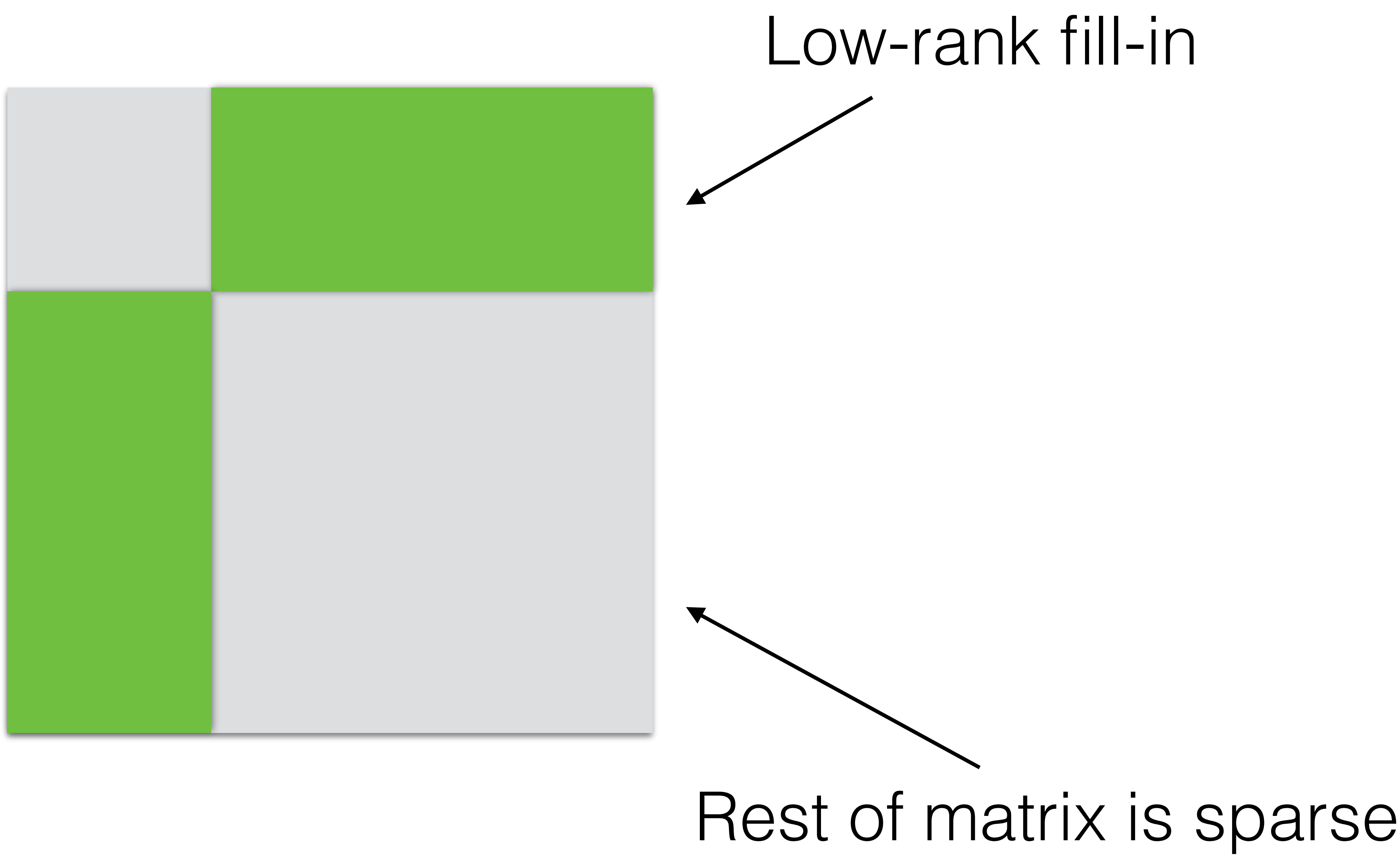
Connection to multigrid

- This method can be connected to multigrid.
- Assume we partition our graph:



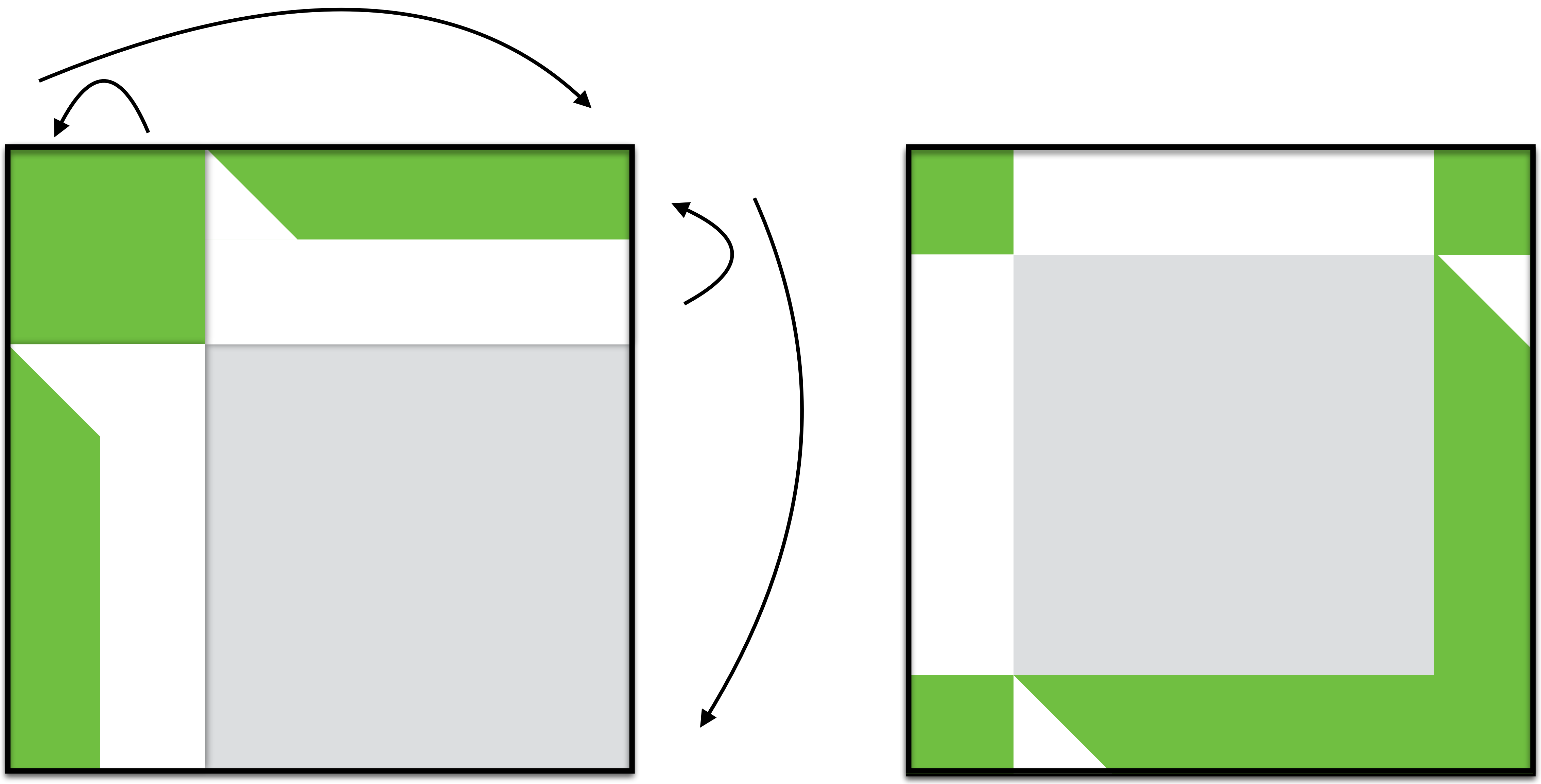
Sparse elimination

- Start a block elimination, following the cluster partitioning shown previously.
- Whenever fill-in occur, we sparsify it.
- What does this mean?


Low-rank fill-in
Rest of matrix is sparse

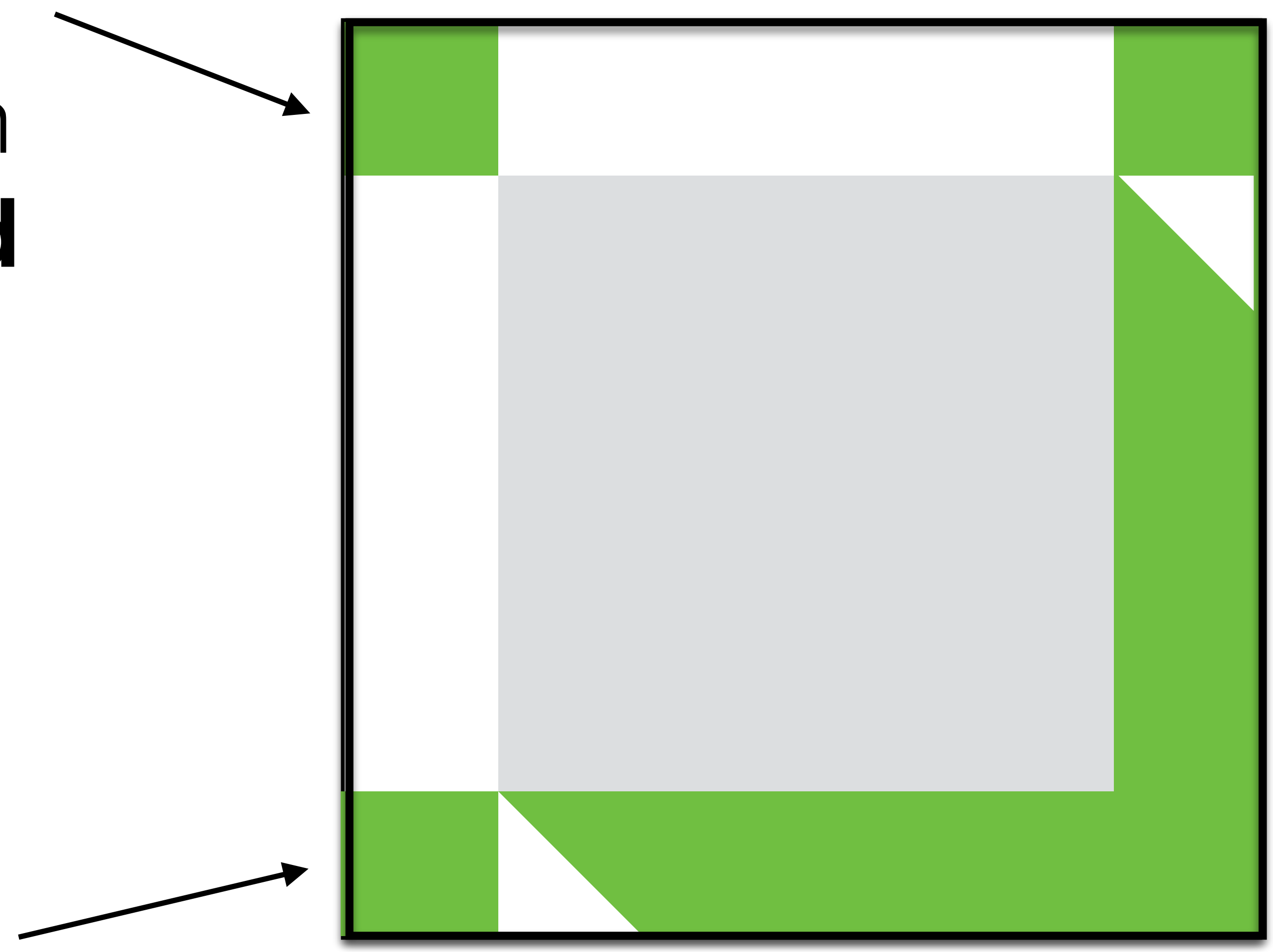
Row and column transformations

Row/Column permutation

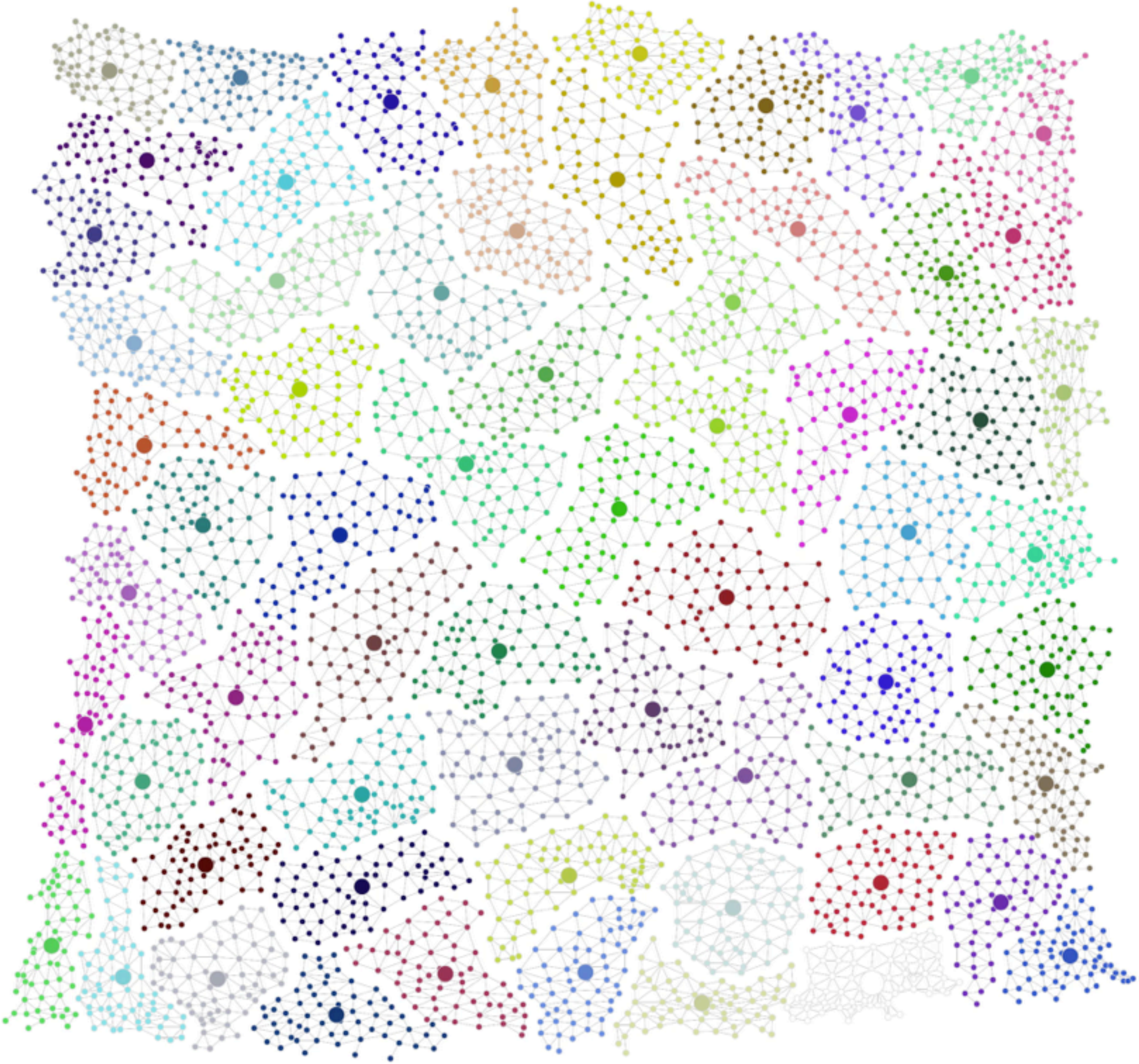


Fine/Coarse

- Elimination of these nodes does not create any new fill-in
- **These are multigrid fine nodes.**



- **These are multigrid coarse nodes.**
- They will be eliminated at the next round.

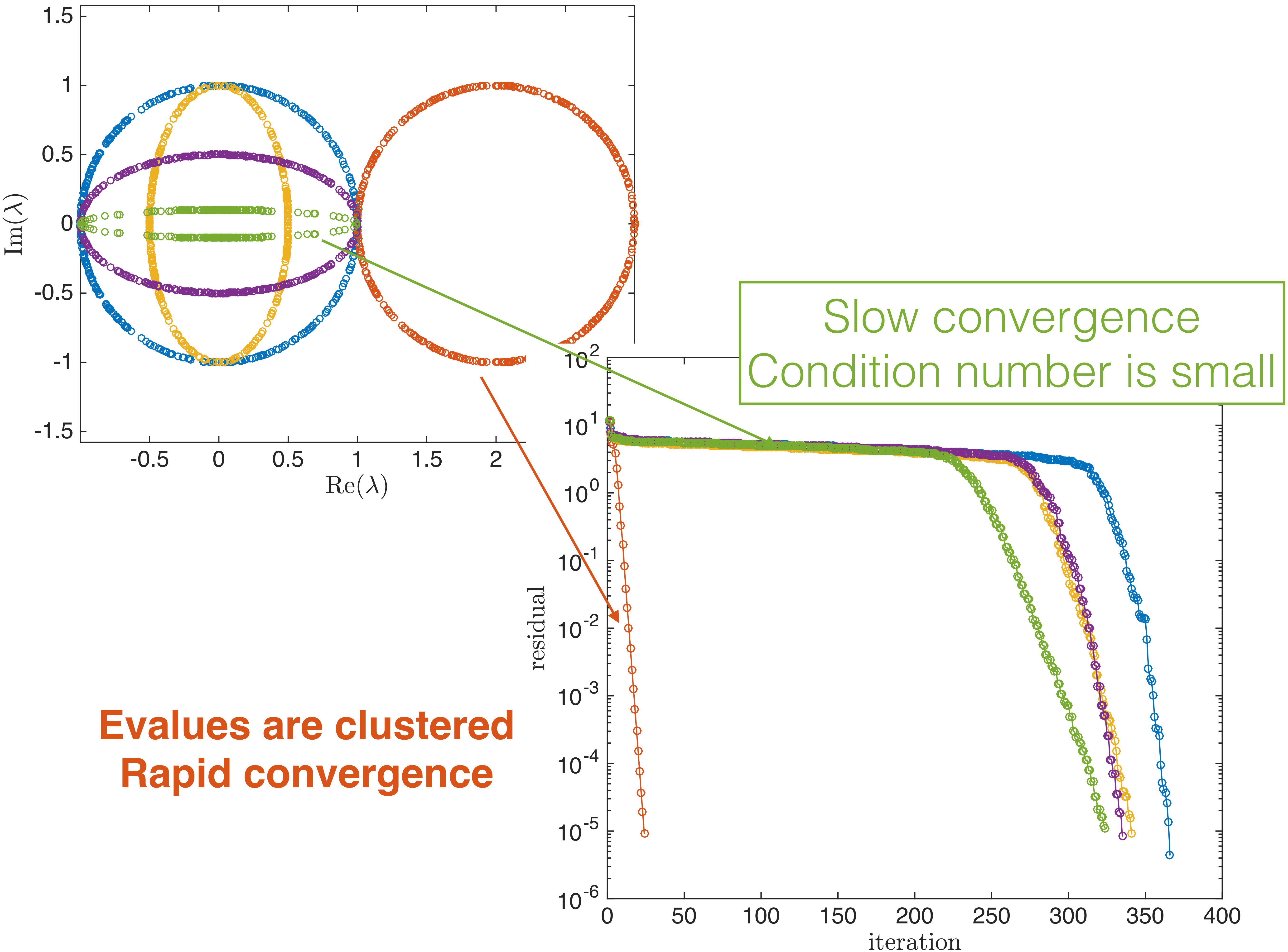


Benchmarks
Convergence of iterative methods

- Examples of convergence behavior.
- For conjugate gradient and symmetric positive definite matrices, the eigenvalues are real and positive. This leads to a simple convergence behavior, based on the condition number.

Unsymmetric systems

- For unsymmetric systems, convergence is more challenging.
- Condition number is still an important factor.
- However, clustering of the eigenvalues is critical.
- An interesting case is eigenvalues distributed on the unit circle.
- The condition number is 1. But convergence is still slow because of the lack of clustering.


Im(λ)
Re(λ)
Slow convergence
Condition number is small
residual
iteration
Evalues are clustered
Rapid convergence

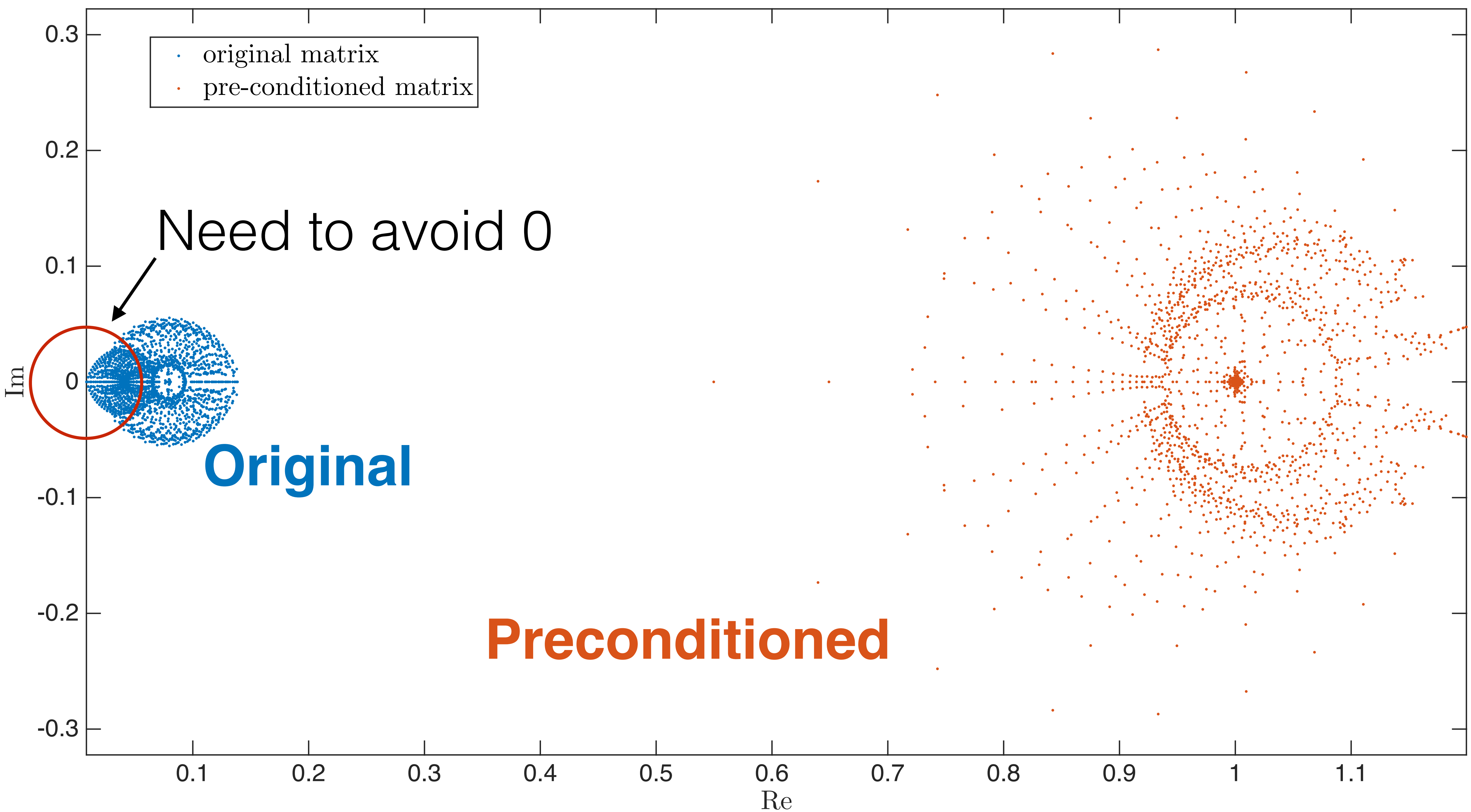
Preconditioning benchmark

- Let's see how this works in practice.
- Radiative transfer equation:

$$\hat{s} \cdot \nabla I + \sigma_e I - \frac{\omega \sigma_e}{4\pi} \int_{4\pi} I d\Omega = (1 - \omega)\sigma_e I_b$$

Unknown: radiation intensity

ILU preconditioning



Boundary element method

- We solve the Helmholtz equation using the boundary element method.

- This uses an integral formulation:

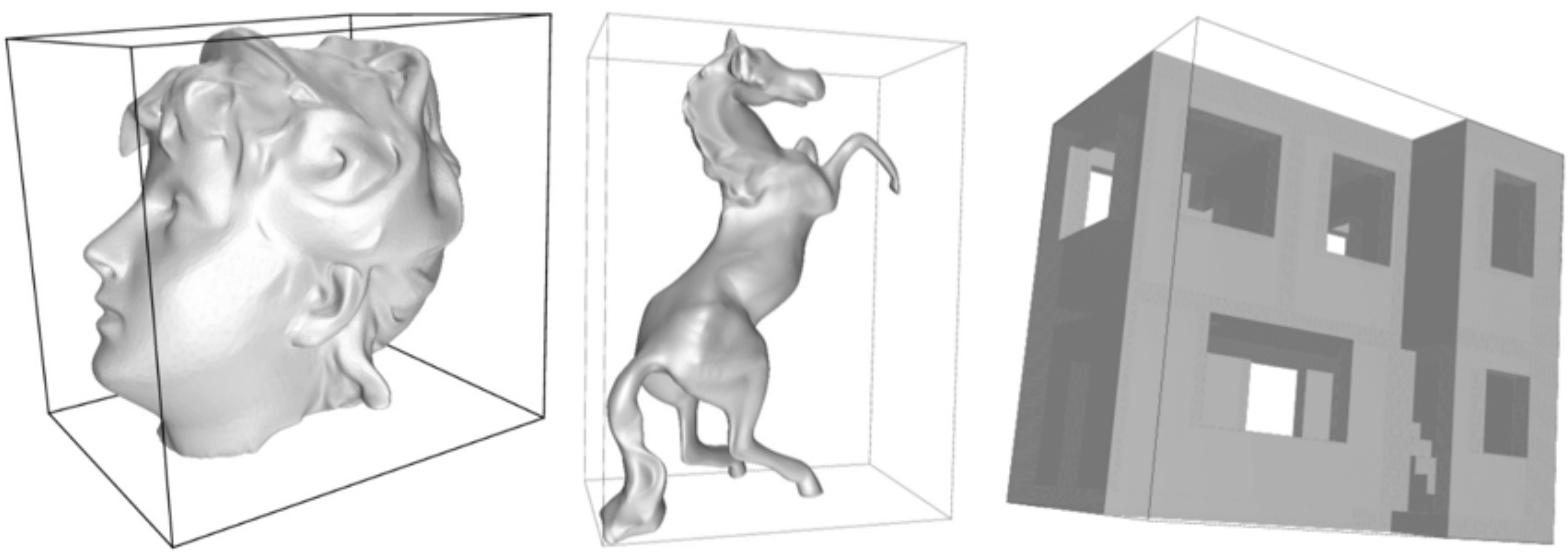
$$\frac{1}{2}u(\boldsymbol{x}) + \int_S \left(\frac{\partial \Gamma}{\partial n_y}(\boldsymbol{x},\boldsymbol{y})u(\boldsymbol{y}) - \Gamma(\boldsymbol{x}-\boldsymbol{y})q(\boldsymbol{y}) \right) \mathrm{d}S_y$$

$$+ \quad \beta \left\{ \frac{1}{2}q(\boldsymbol{x}) + \int_S \left(\frac{\partial^2 \Gamma}{\partial n_x \partial n_y}(\boldsymbol{x},\boldsymbol{y})u(\boldsymbol{y}) - \frac{\partial \Gamma}{\partial n_x}(\boldsymbol{x},\boldsymbol{y})q(\boldsymbol{y}) \right) \mathrm{d}S_y \right\} = u^{\mathrm{I}}(\boldsymbol{x}) + \beta q^{\mathrm{I}}(\boldsymbol{x})$$

- k: wavenumber
- u: pressure field
- $q = \frac{\partial u}{\partial n}$: flux
- $\Gamma(\boldsymbol{x}) = \frac{\exp(ik|\boldsymbol{x}|)}{4\pi|\boldsymbol{x}|}$: fundamental solution of the Helmholtz equation
- $\beta = i/k$: coefficient that makes the integral equation free from fictitious eigenvalues
- u^{I}, $q^{\mathrm{I}} = \frac{\partial u^{\mathrm{I}}}{\partial n}$: incident field

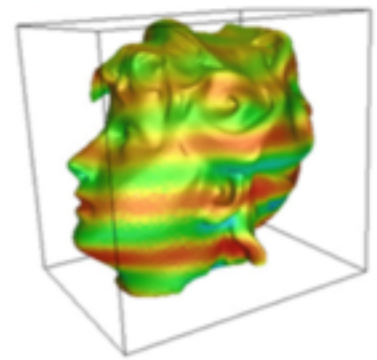
Three geometries w/ Toru Takahashi, Pieter Coulier

Name	Boundary conditions	Incident field	# elements
Head	$q = 0$ (everywhere)	$u^{\mathrm{I}}(\boldsymbol{x}) = \exp(ikx_3)$	$64,944$
Horse	$q = 0$ (everywhere)	$u^{\mathrm{I}}(\boldsymbol{x}) = \exp(ikx_1)$	$190,156$
House	$u = 1$ (on TV), $q = 0$ (everywhere else)	N/A	$147,168$



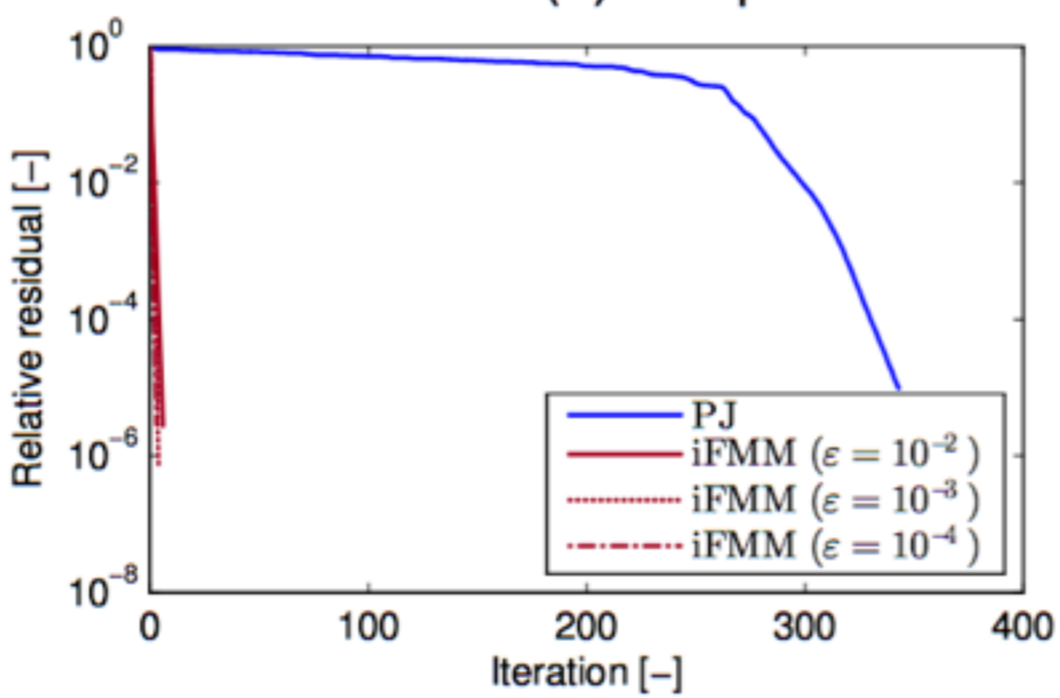
Numerical results: Woman's head

- Sound pressure field $\mathrm{Re}\,(u(\boldsymbol{x}))$ for $k = 32$

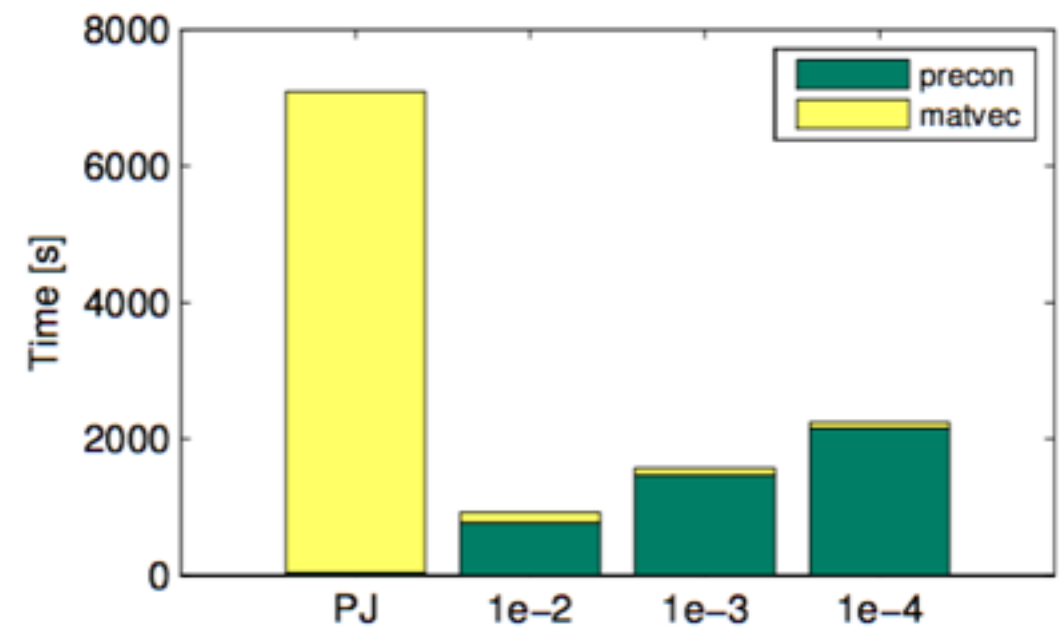


Point Jacobi vs **iFMM (H solver)**

- (a) Relative residual and (b) computation time



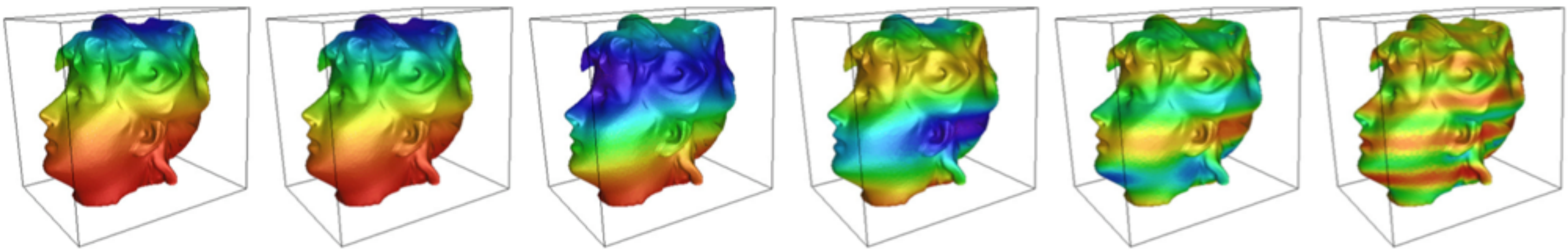
(a)



(b)

PC	# iter	total time [s]	precon. [s]	matvec. [s]	**speed-up**	l_2-error [-]
PJ	343	7091	39	7052		
iFMM ($\varepsilon = 10^{-2}$)	6	922	774	148	**7.7**	2.0×10^{-5}
iFMM ($\varepsilon = 10^{-3}$)	4	1521	1471	104	**4.7**	2.0×10^{-5}
iFMM ($\varepsilon = 10^{-4}$)	3	2242	2158	84	**3.2**	2.0×10^{-5}

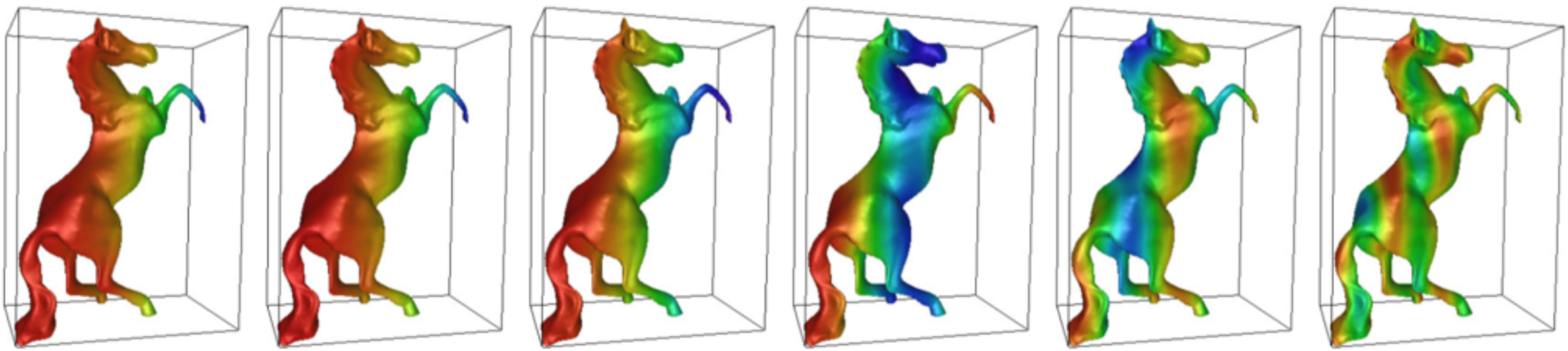
Frequency sweep



k	# iter	total time [s]	precon. [s]	matvec. [s]	**speed-up**	l_2-error [-]
1	91 / 5	1370 / 215	4 / 125	1366 / 90	**6.4**	7.5×10^{-6}
2	86 / 9	1304 / 308	4 / 157	1300 / 151	**4.2**	1.3×10^{-5}
4	77 / 8	1182 / 313	3 / 176	1179 / 137	**3.8**	9.3×10^{-6}
8	88 / 6	1384 / 325	4 / 216	1380 / 109	**4.3**	9.2×10^{-6}
16	147 / 5	2420 / 432	9 / 333	2411 / 99	**5.6**	1.5×10^{-5}
32	343 / 6	7091 / 922	39 / 774	7052 / 148	**7.7**	2.0×10^{-5}

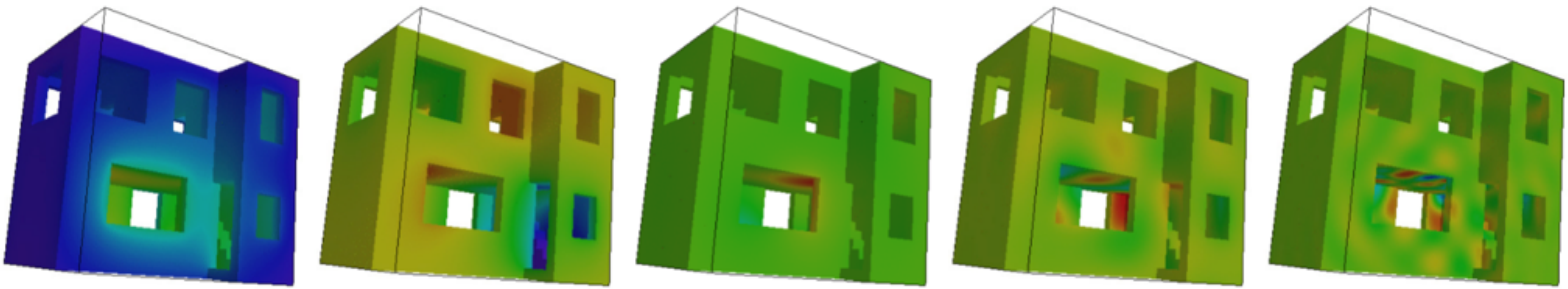
Point Jacobi vs **iFMM (H solver)**

Standing horse



k	# iter	total time [s]	precon. [s]	matvec. [s]	**speed-up**	l_2-error [-]
1	203 / 8	7487 / 572	43 / 245	7444 / 327	**13.1**	1.3×10^{-5}
2	157 / 9	5960 / 632	25 / 268	5935 / 364	**9.4**	1.3×10^{-5}
4	123 / 11	4546 / 794	17 / 353	4529 / 441	**5.7**	9.5×10^{-6}
8	115 / 9	4290 / 754	16 / 384	4274 / 370	**5.7**	1.2×10^{-5}
16	120 / 7	4561 / 728	17 / 426	4544 / 302	**6.3**	1.1×10^{-5}
32	185 / 9	7553 / 1155	38 / 748	7515 / 407	**6.5**	1.3×10^{-5}

TV in the living room



k	# iter	total time [s]	precon. [s]	matvec. [s]	**speed-up**	l_2-error [-]
1	90 / 6	1869 / 419	7 / 275	1861 / 144	**4.5**	2.9×10^{-3}
2	140 / 5	2921 / 453	16 / 329	2905 / 124	**6.4**	1.3×10^{-3}
4	269 / 5	5777 / 695	45 / 567	5732 / 128	**8.3**	1.1×10^{-2}
8	583 / 10	13673 / 1378	189 / 1123	13484 / 255	**9.9**	2.3×10^{-3}
16	1384 / 19	45839 / 3389	1008 / 2733	44831 / 656	**13.5**	2.8×10^{-3}

Indefinite systems w/ Kai Yang

$$\triangle u - \lambda u = f$$

- No good preconditioner exists for these problems.
- **ILU and MG/AMG fail for these matrices.**
- λ is chosen from the interval $[\lambda_{min}, \lambda_{max}]$ for the Laplacian.
- 2D Poisson with 10k points.

Convergence of H solver

Problem	setup time (s)	solve time(s)	number of iterations
A1	0.46	0.09	9
A2	0.56	0.21	18
A3	0.65	0.2	16
A4	0.72	0.14	11
A5	0.7	0.14	11
A6	0.64	0.25	20
A7	0.56	0.18	15
A8	0.46	0.11	10

Various eigenvalue shifts

Frequency sweep

Mid
$5\ \lambda$ →

High
$40\ \lambda$ →

# of unknowns	tree depth	ϵ	setup (s)	solve (s)	# of iterations	largest size of red node
2.5k	7	1e-3	0.07	0.03	24	19
		1e-4	0.1	0.01	8	19
		1e-5	0.1	0.01	4	19
		1e-6	0.1	0.01	3	19
10k	9	1e-4	0.62	0.2	17	28
		1e-5	0.64	0.07	6	29
		1e-6	0.66	0.05	4	33
40k	11	1e-5	4.49	0.57	9	70
		1e-6	4.66	0.46	7	71
160k	13	1e-6	35.23	12.63	41	112

Software sample

- w/ Hadi Pouransari:
  https://bitbucket.org/hadip/lorasp
  Lorasp: hierarchical linear solver for sparse matrices.

- w/ Pieter Coulier: hierarchical linear solver for dense matrices; iFMM. Requires an FMM formulation (e.g., BEM, integral equation)

- w/ Toru Takahashi: fast Helmholtz solver using hierarchical matrices.

References

- Fast hierarchical solvers for sparse matrices using low-rank approximation, Hadi Pouransari, Pieter Coulier, Eric Darve; arXiv: 1510.07363,
  http://arxiv.org/abs/1510.07363

- The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems; Pieter Coulier, Hadi Pouransari, Eric Darve; arXiv:1508.01835
  http://arxiv.org/abs/1508.01835

- Aminfar, A., and E. Darve. “A fast, memory efficient and robust sparse preconditioner based on a multifrontal approach with applications to finite-element matrices.” *Int. J. Num. Meth. Eng.* (2016): doi 10.1002/nme.5196

Hands-on

- Log on
  https://juliabox.org/
- Run sample code to see that everything works for you.

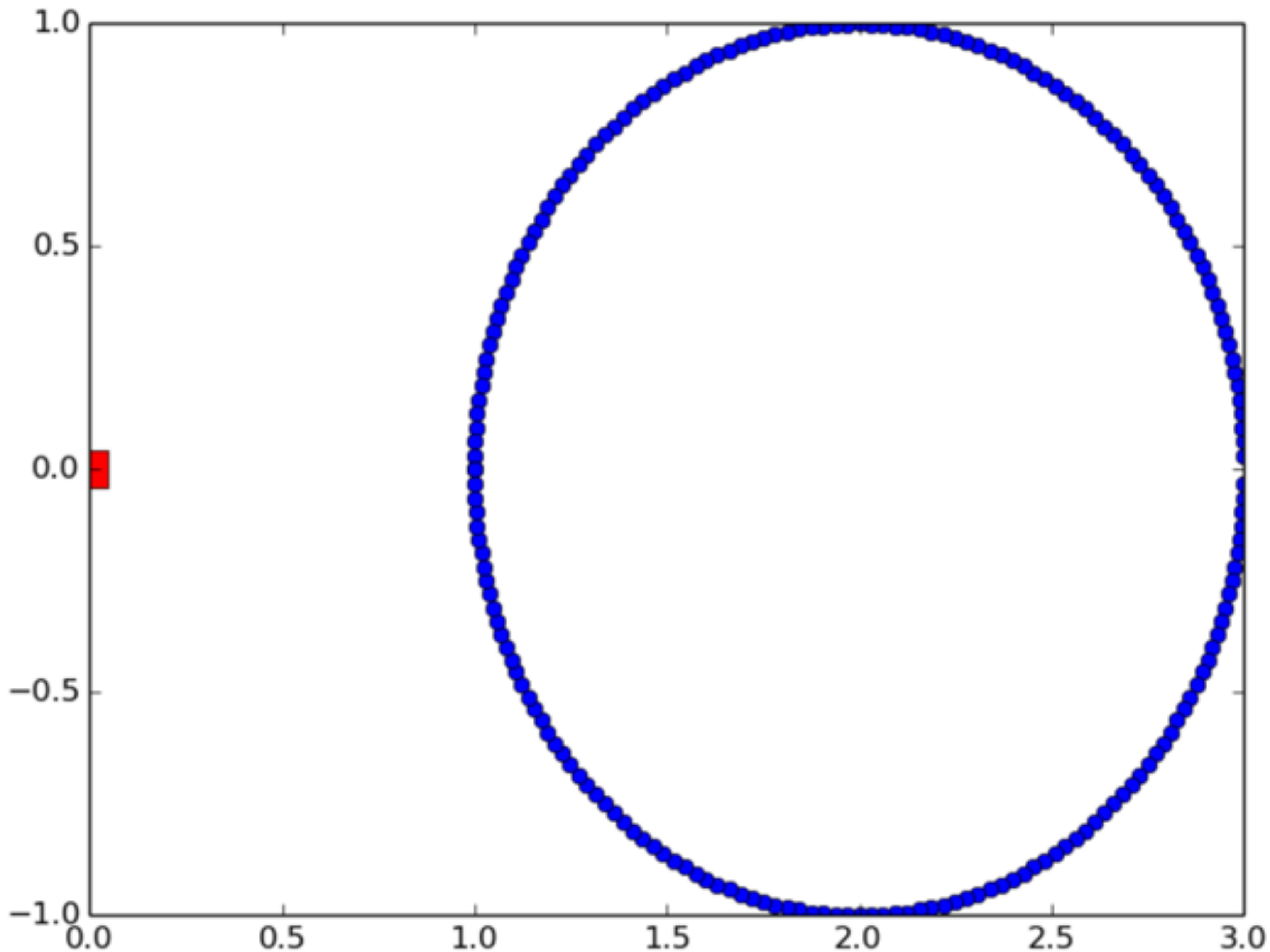
Lab 1: convergence of iterative solvers

- We create matrices with different eigenvalue distributions.
- The linear system is solved using GMRES.

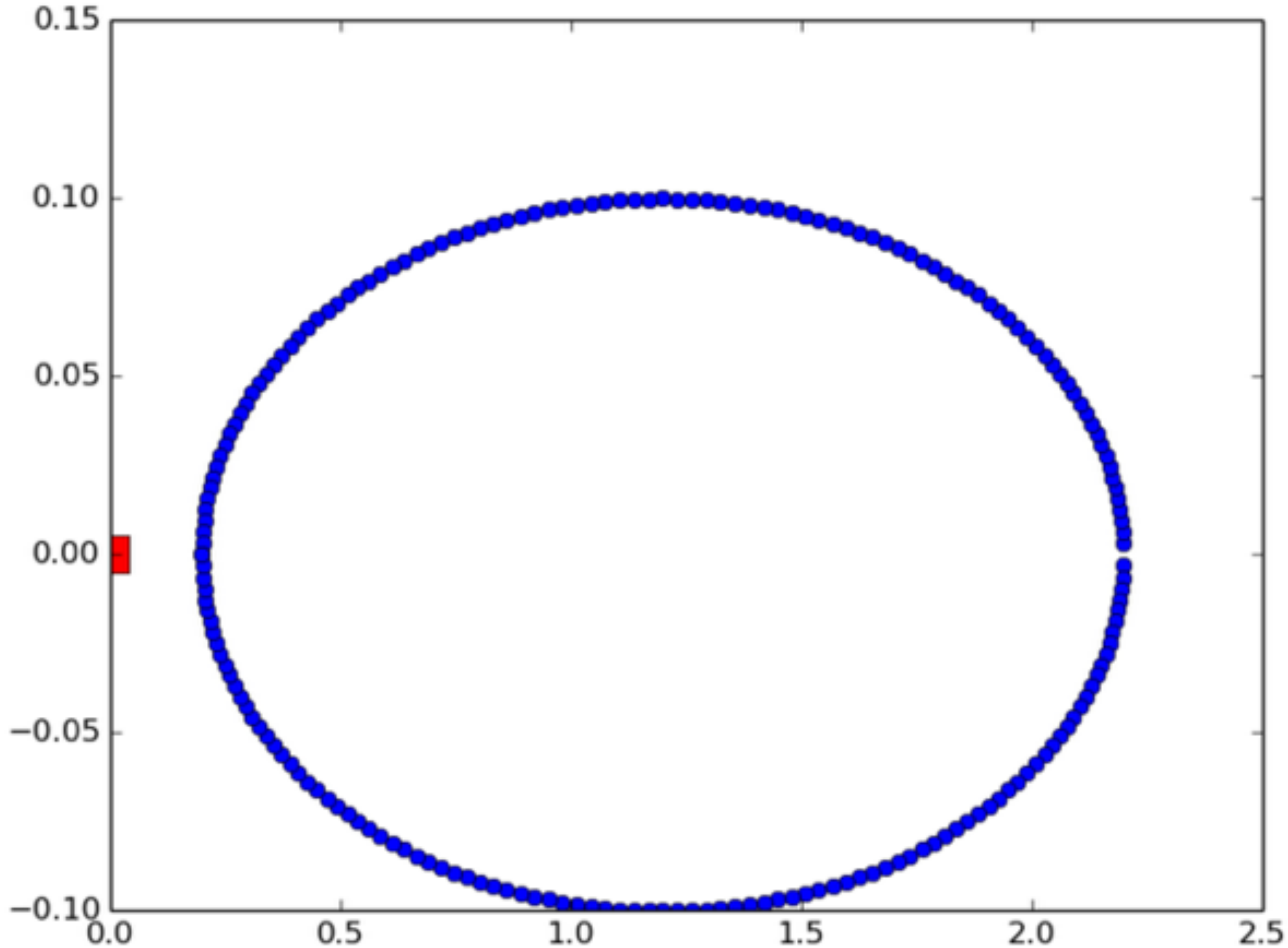
Eigenvalue distributions

- Try out these different cases.
- What do you observe? How fast is the convergence? Can you explain your observations?

Distribution #1

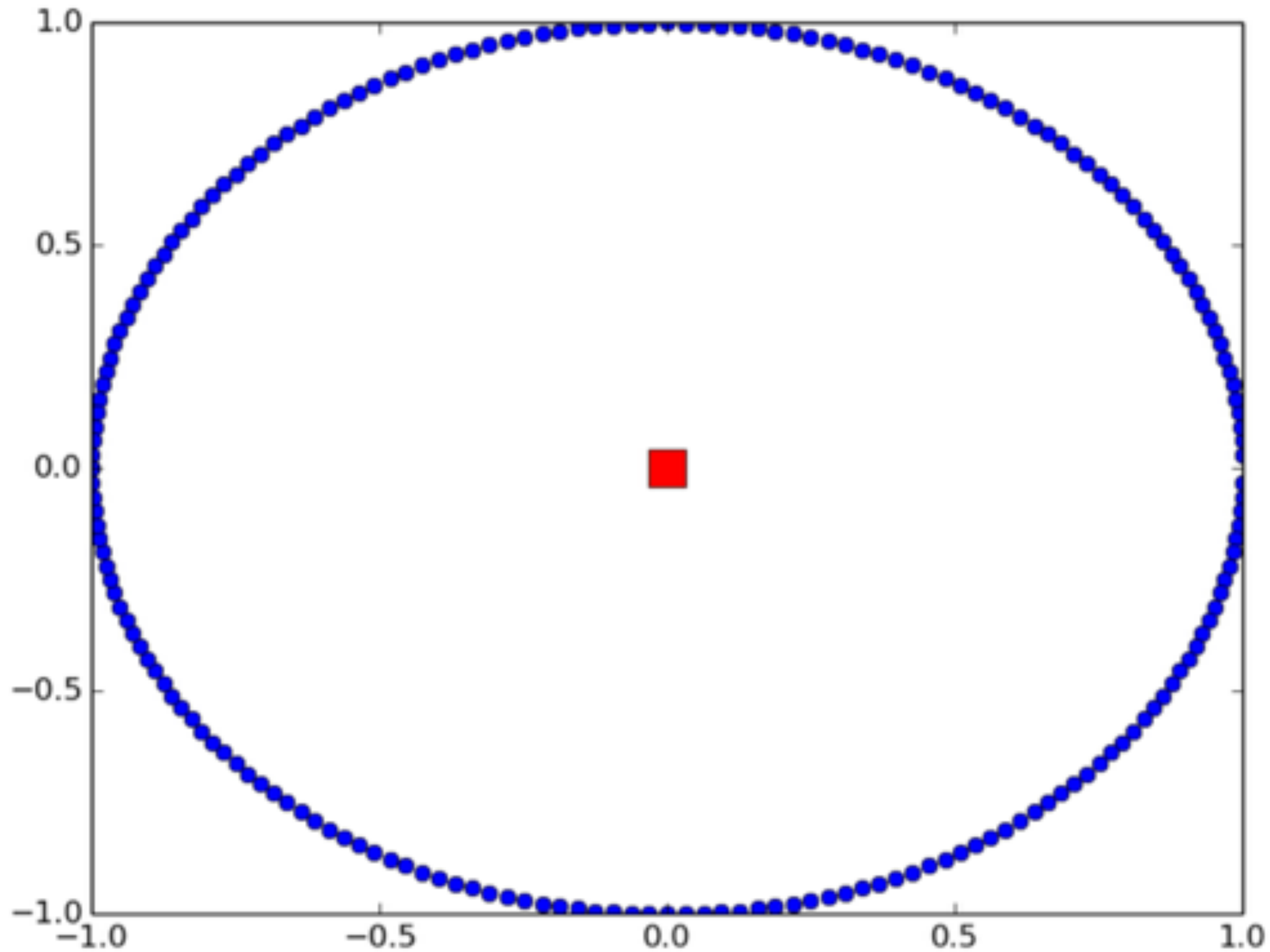


Distribution #2



Can you make GMRES convergence very slowly by changing x_shift?

Distribution #3



This matrix corresponds to rotations in different planes.

Try playing around with other eigenvalue distributions!

Hierarchical Matrices

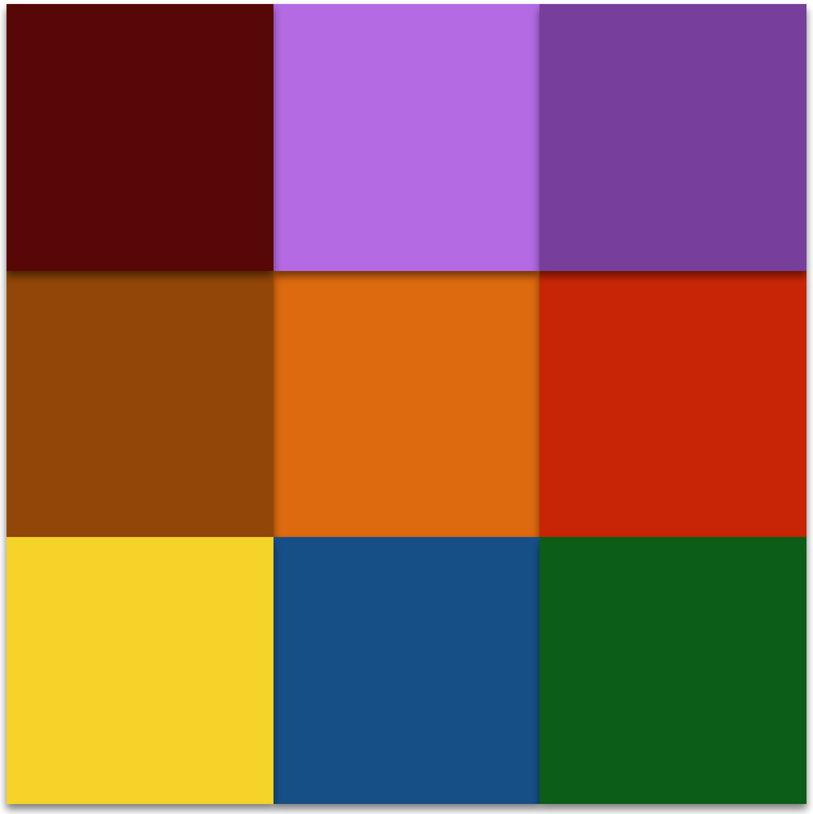
- One fundamental property we use in hierarchical matrix calculation is that the Schur complement can be compressed during an LU/Cholesky factorization.
- Is that true in practice?
- What types of PDE satisfy this compression property?
- Let's investigate.

PDE solver

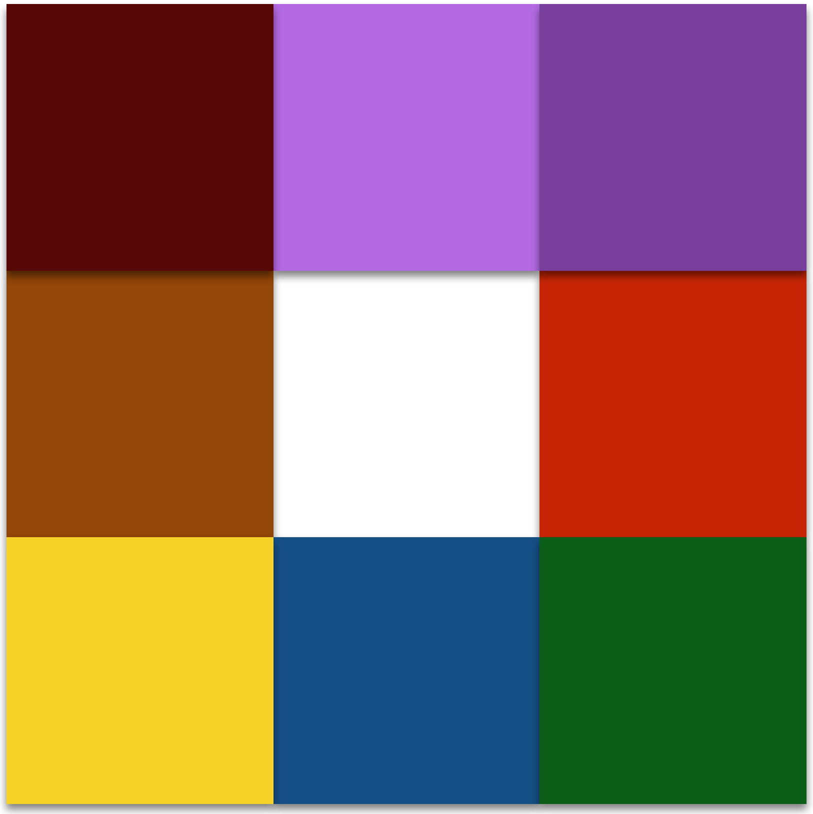
- Consider a regular mesh and a 5 point stencil for:

$$-k^2\,T + e \cdot \nabla T - D\nabla^2 T = \mathrm{RHS}$$

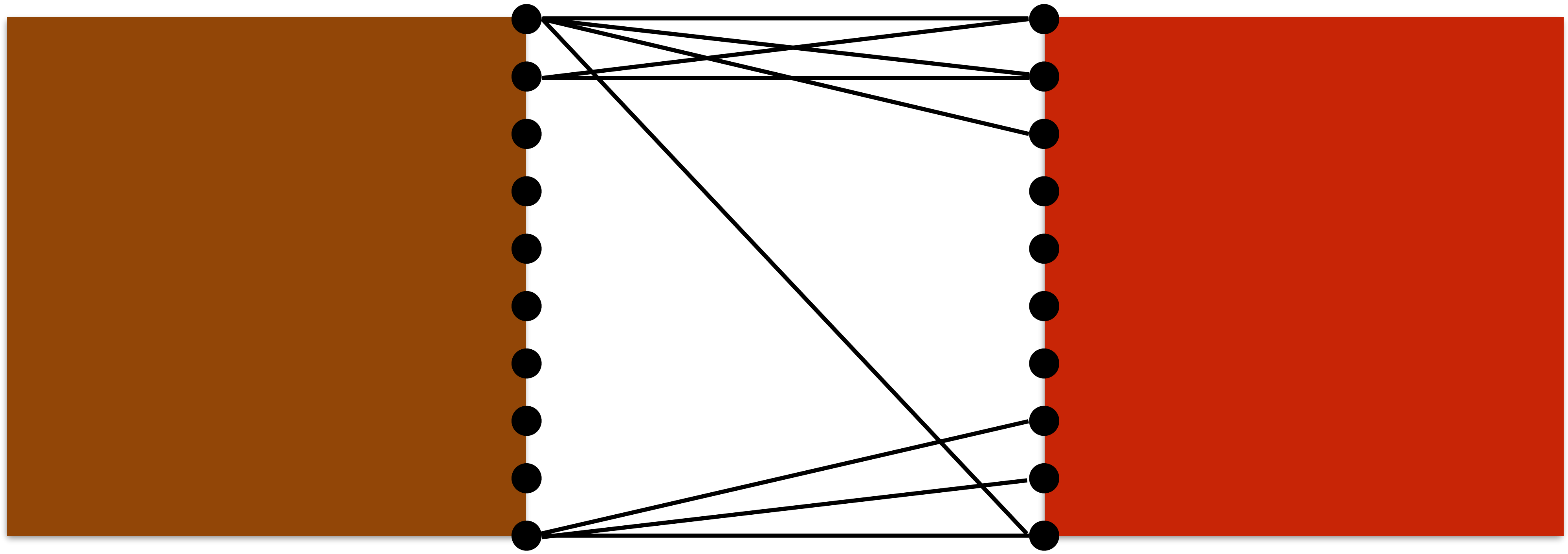
- Let’s do a Gaussian elimination (e.g., LU) on some part of the grid.



Schematic view of a 2D grid,
partitioned into 9 subdomains



- Eliminate the center domain of the grid
- LU factorization where we eliminate rows & columns associated with the center domain

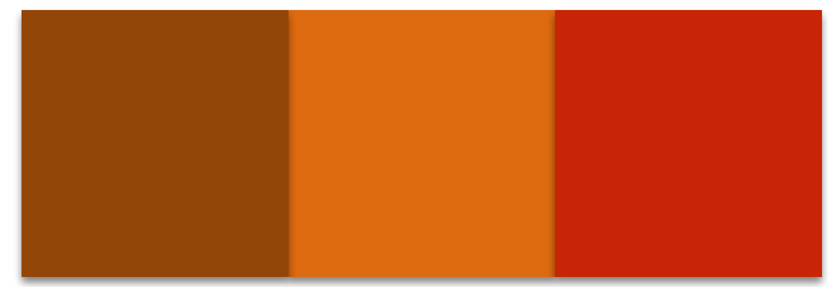


$$A = \begin{pmatrix} I & \\ UA_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & \\ 0 & A_{22} - UA_{11}^{-1}U^T \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1}U^T \\ & I \end{pmatrix}$$

- Points on the left and right boundaries become all connected.
- This forms a **dense block** in the matrix.
- A key assumption in Hierarchical Solvers is that this matrix must have low-rank blocks.
- Is that in fact the case?

Set up of benchmark

- Matrix of system, focusing on the 3 clusters, in the middle row:



$$\begin{pmatrix} A_{CC} & A_{CL} & A_{CR} \\ A_{LC} & A_{LL} & 0 \\ A_{RC} & 0 & A_{RR} \end{pmatrix}$$

C: center; L: left; R: right

- Let's eliminate A_{CC}

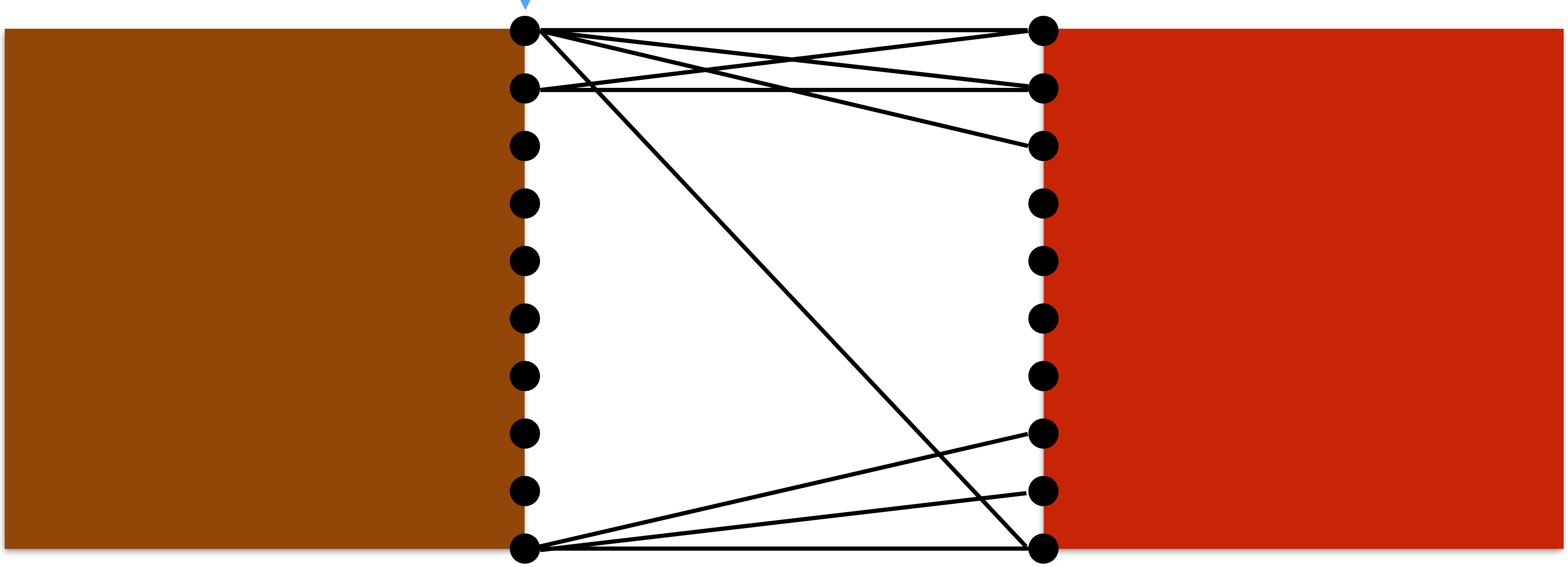
$$\begin{pmatrix} A_{CC} & A_{CL} & A_{CR} \\ A_{LC} & A_{LL} & 0 \\ A_{RC} & 0 & A_{RR} \end{pmatrix}$$

Fill-in

$$\begin{pmatrix} A_{LL} - A_{LC}A_{CC}^{-1}A_{CL} & -A_{LC}A_{CC}^{-1}A_{CR} \\ -A_{RC}A_{CC}^{-1}A_{CL} & A_{RR} - A_{RC}A_{CC}^{-1}A_{CR} \end{pmatrix}$$

Modified self-interaction

Fill-in between L and R



Low-rank assumption

$$\begin{pmatrix} A_{LL} - A_{LC}A_{CC}^{-1}A_{CL} & -A_{LC}A_{CC}^{-1}A_{CR} \\ -A_{RC}A_{CC}^{-1}A_{CL} & A_{RR} - A_{RC}A_{CC}^{-1}A_{CR} \end{pmatrix}$$

- For hierarchical solvers to be efficient, this block should be low-rank.

- Let’s test this.

Case #1

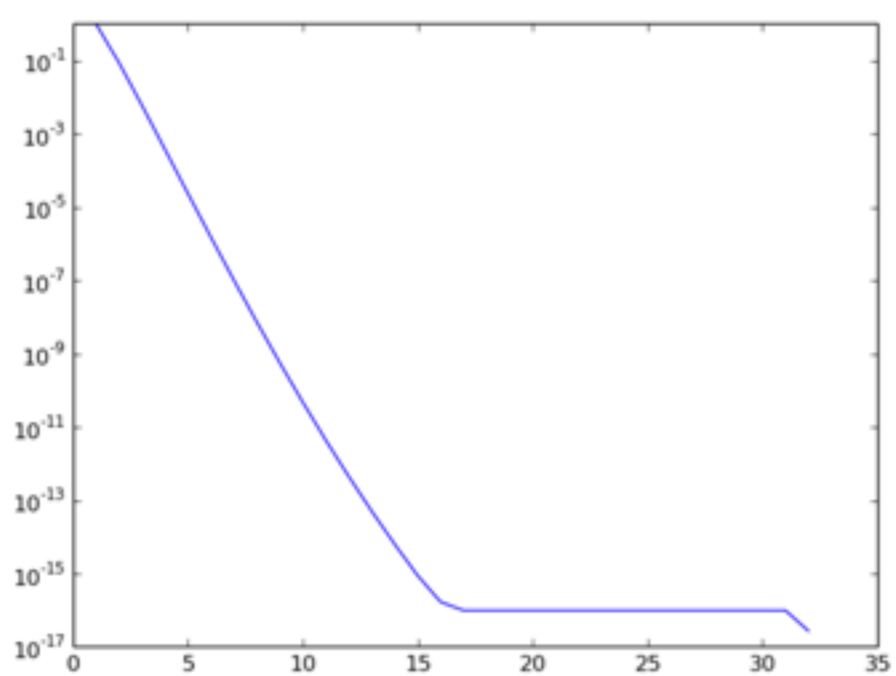
Pure diffusion equation.

```
k  = 0          # shift

D  = 1          # diffusion

ex = 0          # velocities

ey = 0
```



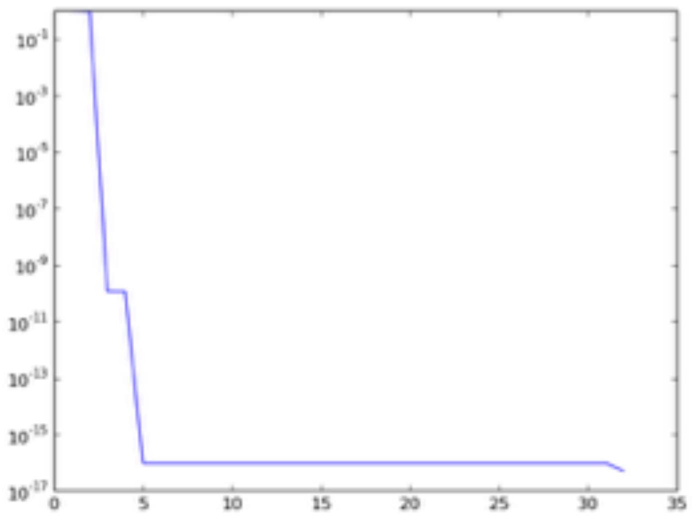
Case #2

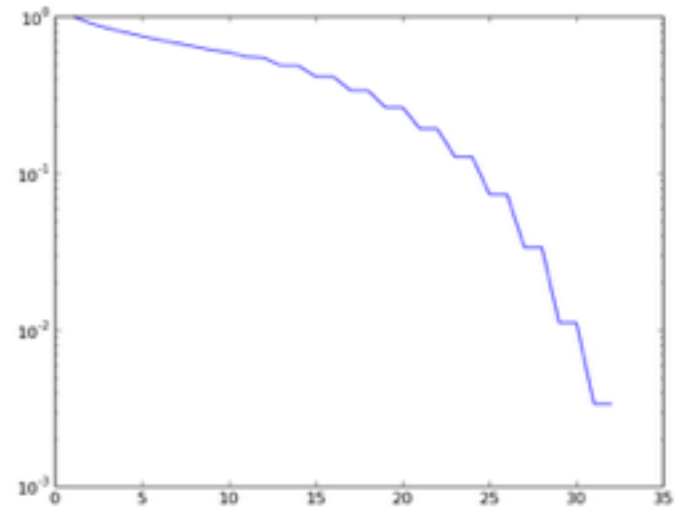
Convection dominated

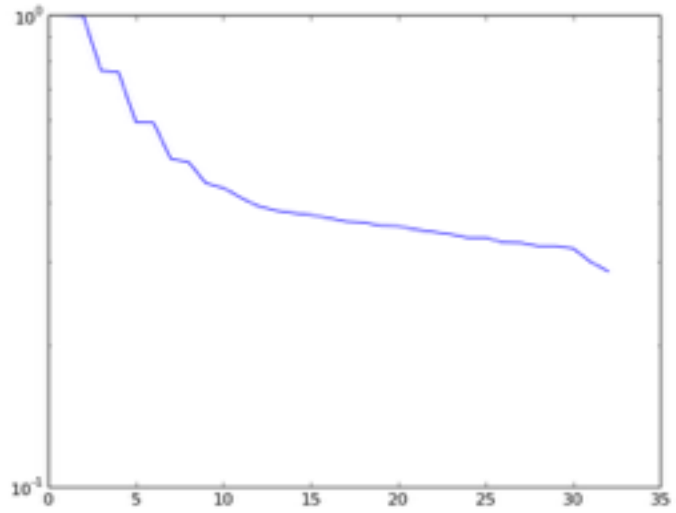
```
k  = 0          # shift

D  = 0.01       # diffusion
```

ex	0.1	1	1
ey	1	1	0.1







Case #3

Oscillatory system

```
D  = 1       # diffusion

ex = ey = 0 # velocity
```

k 0.1 0.5 2.5