H-matrix techniques for parallel hybrid solvers

Yuval HARNESS - INRIA Team HiePACS
Joint work with
Emmanuel AGULLO, Luc GIRAUD (INRIA) & Eric DARVE (Stanford)

CIMI HPC Semester, ENSEEIHT, Toulouse, 24-26 June 2015

Outline

Background

Goal: Solve large sparse linear problems $Ax = b$
A and b are given with no a-priori information on A.

Hybrid Linear Solver

- Employs standard domain decomposition methodology
- Takes advantage of both direct and iterative approaches

This Talk

- MaPHyS (Massively Parallel Hybrid Solver)
- Hierarchical matrix techniques for preconditioning

MaPHyS/Algbraic Domain Decomposition

Input: sparse linear system , $Ax = b$.

MaPHyS/Algbraic Domain Decomposition

Input: sparse linear system , $Ax = b$.
$\rightarrow$ algebraic domain decomposition:

$$Ax = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix}$$

$A_{II} \leftrightarrow$ interior subdomains, $A_{\Gamma\Gamma} \leftrightarrow$ separators.

MaPHyS/Algbraic Domain Decomposition

Input: sparse linear system , $Ax = b$.
$\rightarrow$ algebraic domain decomposition:

$$Ax = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix}$$

$A_{II} \leftrightarrow$ interior subdomains, $A_{\Gamma\Gamma} \leftrightarrow$ separators.

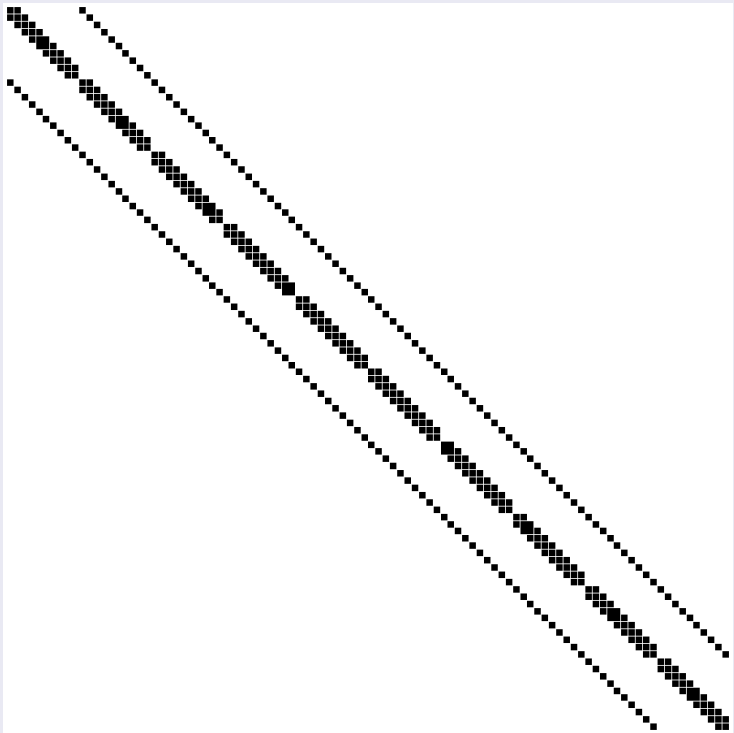
$$A = \left(\begin{array}{cccc|c} A_{I_1 I_1} & & & & A_{I_1\Gamma} \\ & A_{I_2 I_2} & & & A_{I_2\Gamma} \\ & & \ddots & & \vdots \\ & & & A_{I_p I_p} & A_{I_p\Gamma} \\ \hline A_{\Gamma I_1} & A_{\Gamma I_2} & \cdots & A_{\Gamma I_p} & A_{\Gamma\Gamma} \end{array}\right)$$

MaPHyS/Algbraic Domain Decomposition

Method: Partitioning the adjacency graph,

$$G = (\{1, \ldots, n\}, \{(i,j), \ A_{ij} \neq 0 \mid A_{ji} \neq 0\})$$

Matrix A

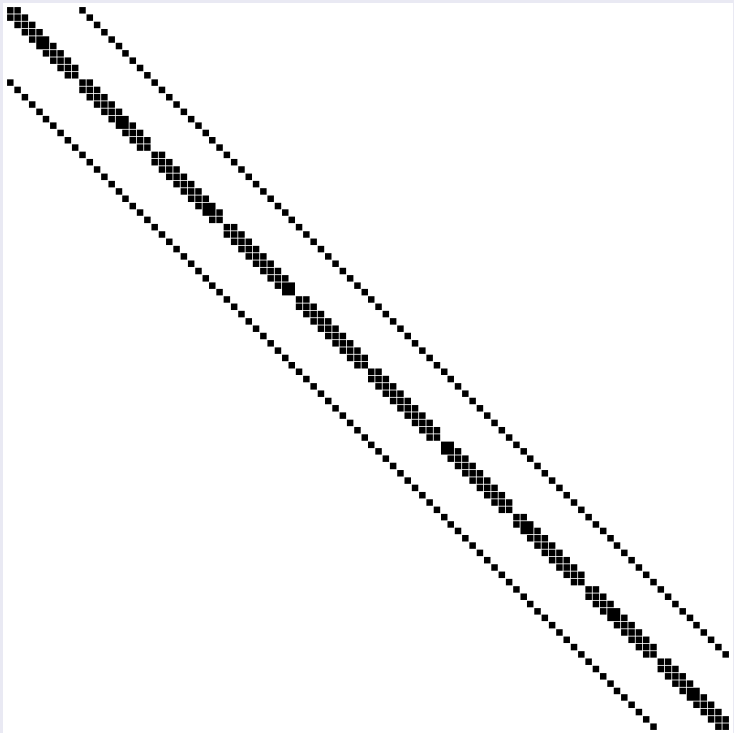


MaPHyS/Algbraic Domain Decomposition

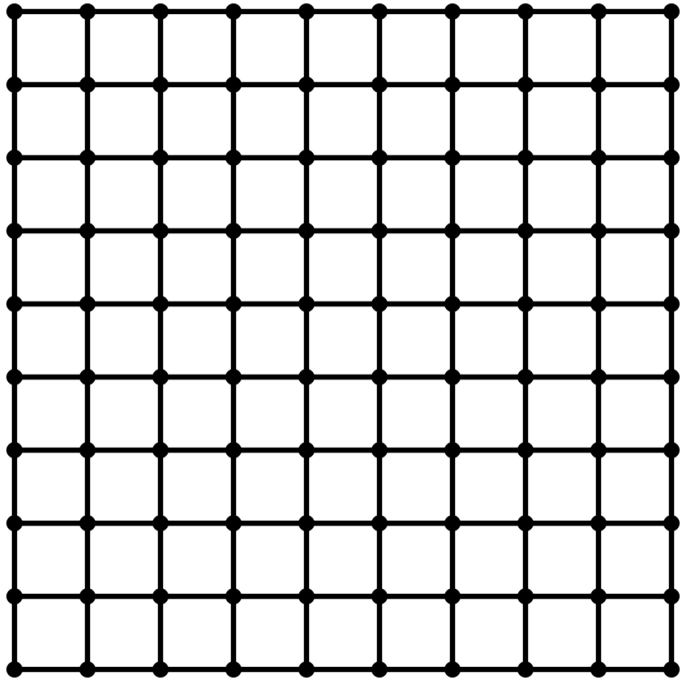
Method: Partitioning the adjacency graph,

$$G = (\{1, \ldots, n\}, \{(i,j), \ A_{ij} \neq 0 \mid A_{ji} \neq 0\})$$

Matrix A



Adjacency graph G

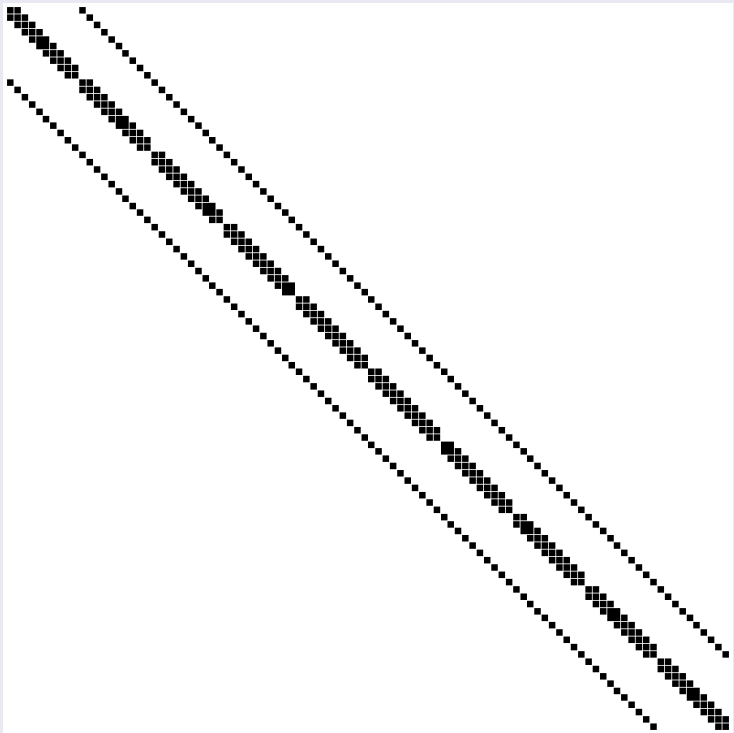


MaPHyS/Algbraic Domain Decomposition

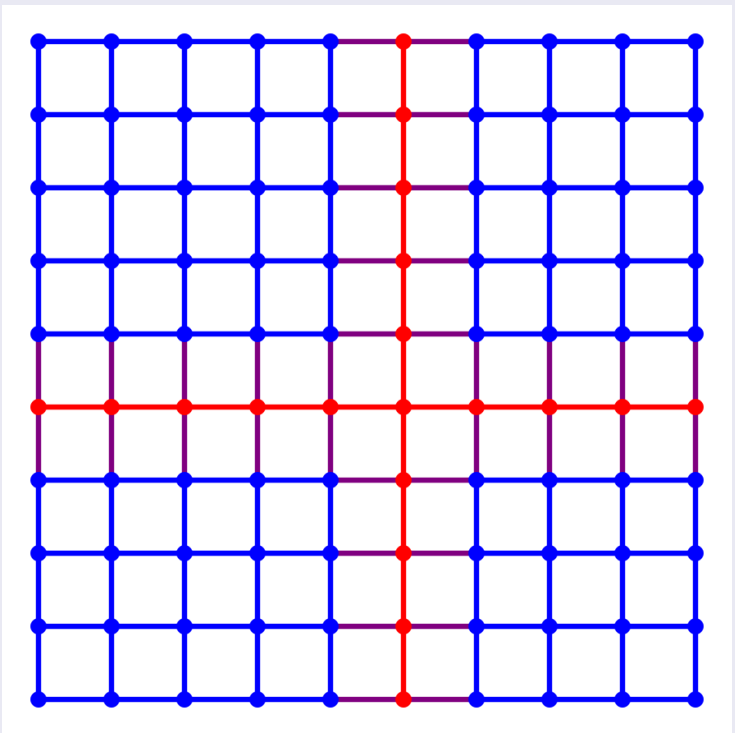
Method: Partitioning the adjacency graph,

$$G = (\{1, \ldots, n\}, \{(i,j), \ A_{ij} \neq 0 \mid A_{ji} \neq 0\})$$

Matrix A



Adjacency graph G

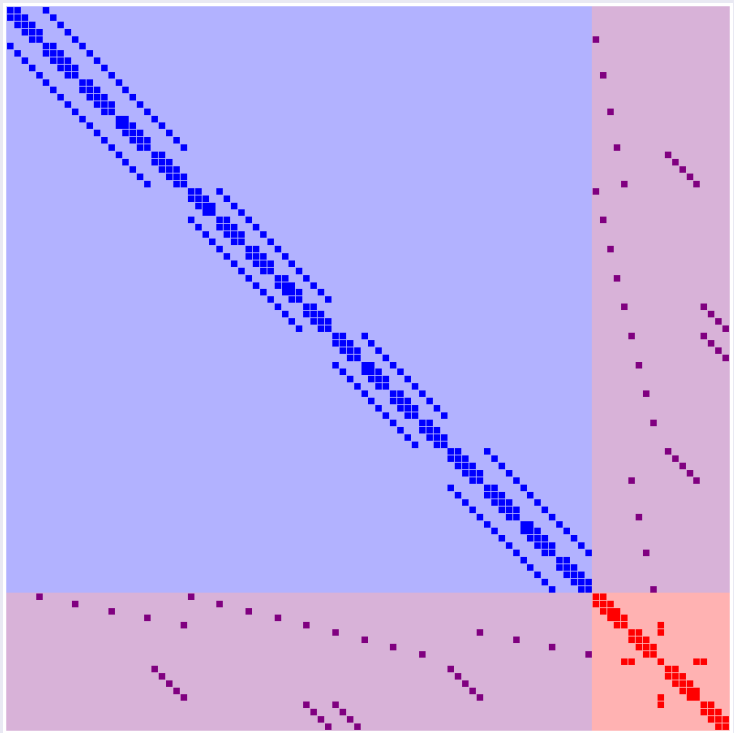


MaPHyS/Algbraic Domain Decomposition

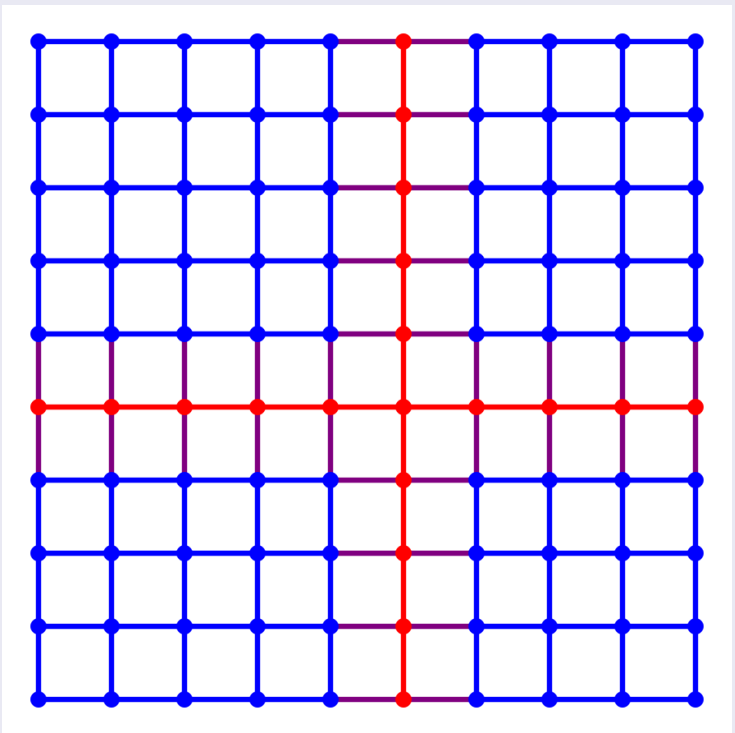
Method: Partitioning the adjacency graph,

$$G = (\{1, \ldots, n\}, \{(i,j), \ A_{ij} \neq 0 \mid A_{ji} \neq 0\})$$

Matrix A



Adjacency graph G

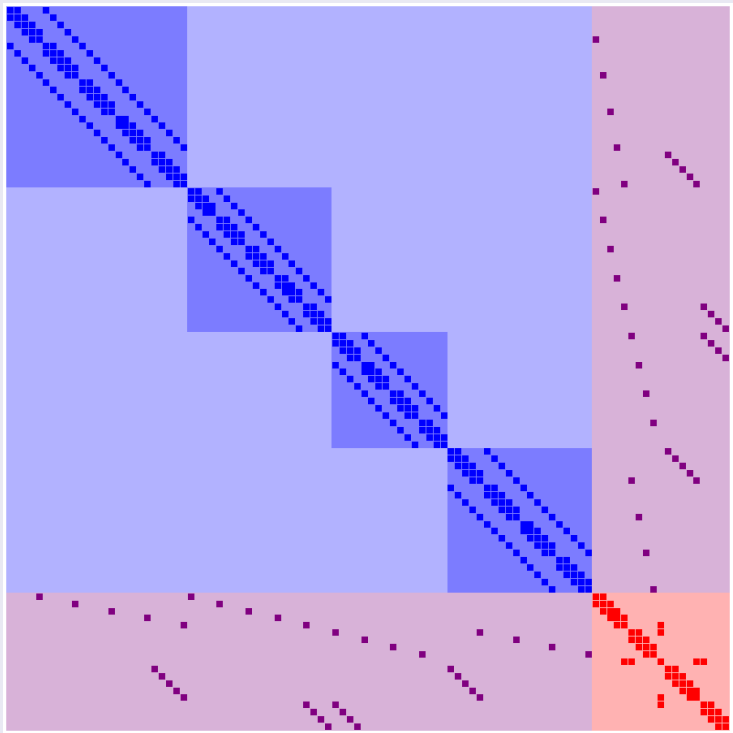


MaPHyS/Algbraic Domain Decomposition

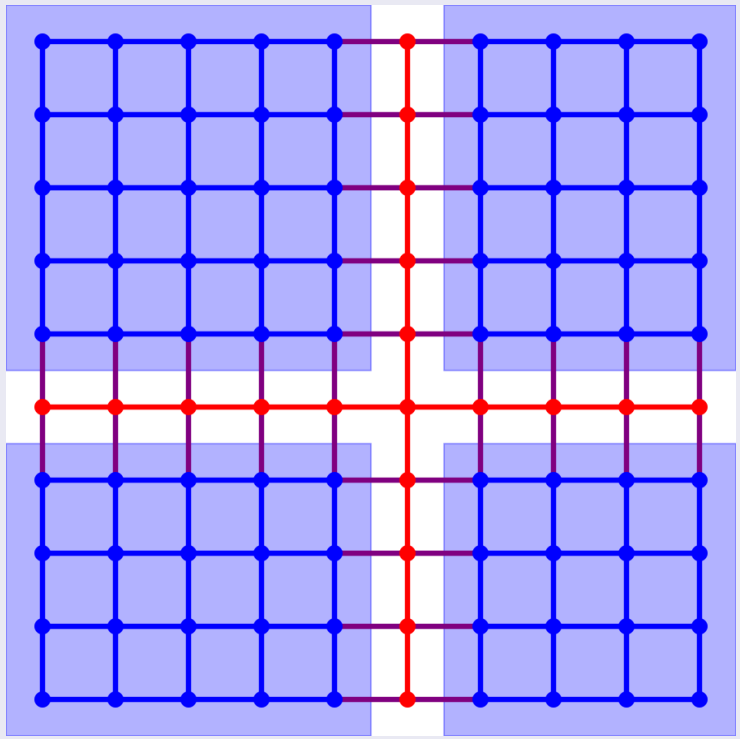
Method: Partitioning the adjacency graph,

$$G = (\{1, \ldots, n\}, \{(i,j), \ A_{ij} \neq 0 \mid A_{ji} \neq 0\})$$

Matrix A



Adjacency graph G



MaPHyS/The Schur System

The (Global) Schur System:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix} \quad \Leftrightarrow \quad \begin{array}{rcl} x_I & = & A_{II}^{-1}\left(b_I - A_{I\Gamma} x_\Gamma\right) \\ x_\Gamma & = & S^{-1} b_\Gamma \end{array}$$

MaPHyS/The Schur System

The (Global) Schur System:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix} \Leftrightarrow \begin{array}{rcl} x_I & = & A_{II}^{-1}\left(b_I - A_{I\Gamma}x_\Gamma\right) \\ x_\Gamma & = & S^{-1}b_\Gamma \end{array}$$

The Schur Complement:

$$S = A_{\Gamma\Gamma} - A_{\Gamma I}A_{II}^{-1}A_{I\Gamma}$$

MaPHyS/The Schur System

The (Global) Schur System:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix} \quad \Leftrightarrow \quad \begin{array}{rcl} x_I & = & A_{II}^{-1}\left(b_I - A_{I\Gamma} x_\Gamma\right) \\ x_\Gamma & = & S^{-1} b_\Gamma \end{array}$$

The Schur Complement:

$$S = A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{I\Gamma}$$

- $Sx_\Gamma = b_\Gamma$ is solved iteratively.
- $\kappa(S) < \kappa(A)$.
- S is never formed, but assembled at each iteration:

$$Sx_\Gamma = \sum_{i=1}^{N} R_i^T S_i R_i x_\Gamma \quad : \quad R_i : \mathbb{R}^n \to \mathbb{R}^{n_i}$$

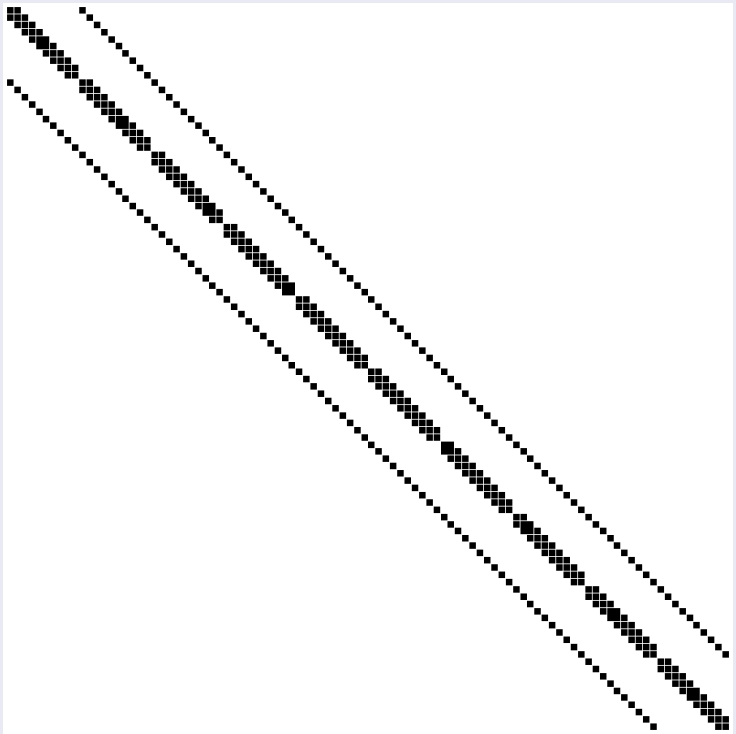
- All the local components are computed in parallel.

MaPHyS/Local Components

Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \quad \Rightarrow \quad S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A

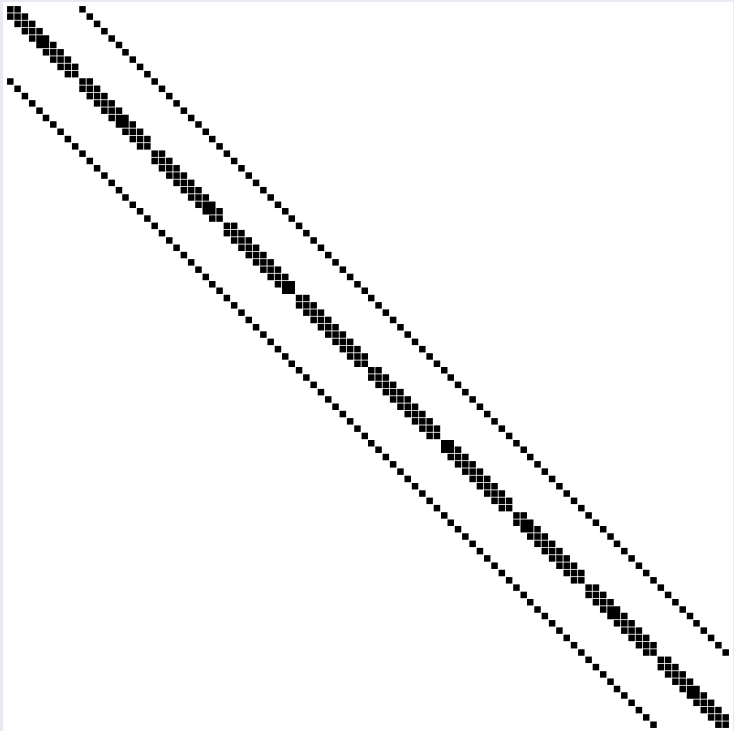


MaPHyS/Local Components

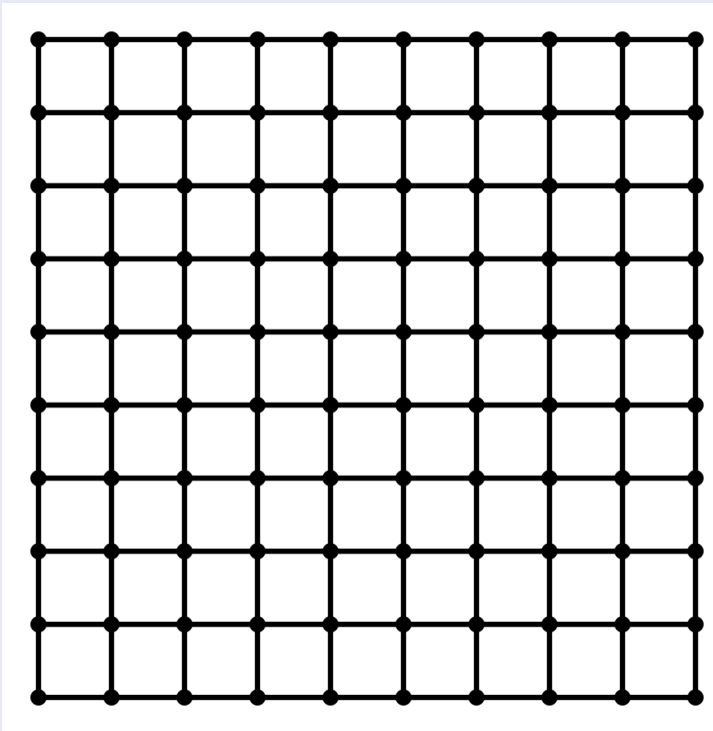
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A



Adjacency graph G

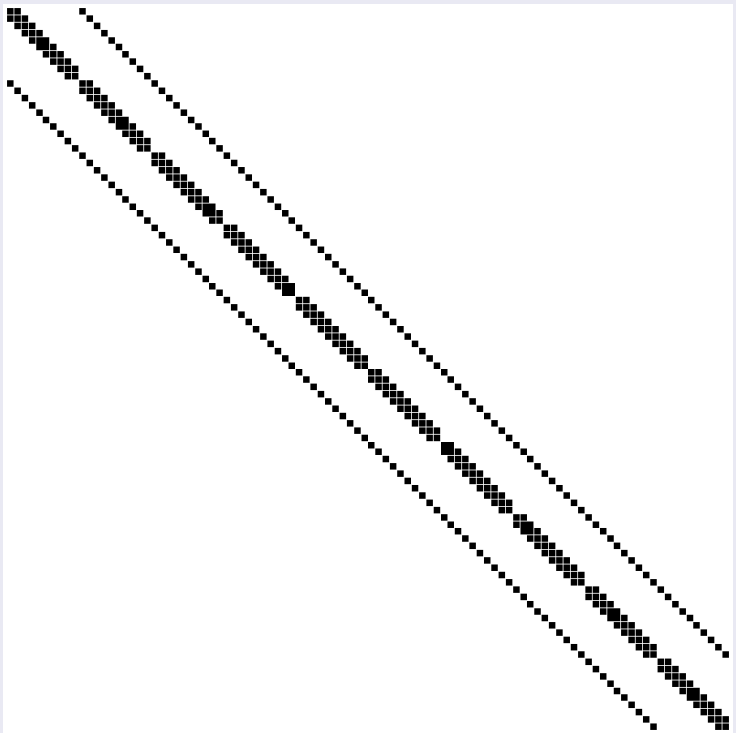


MaPHyS/Local Components

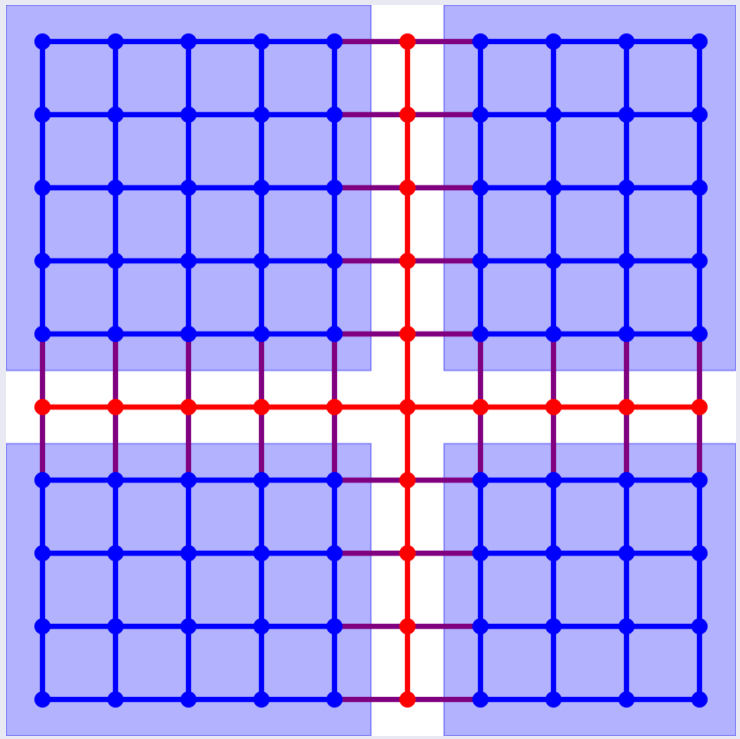
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A



Adjacency graph G

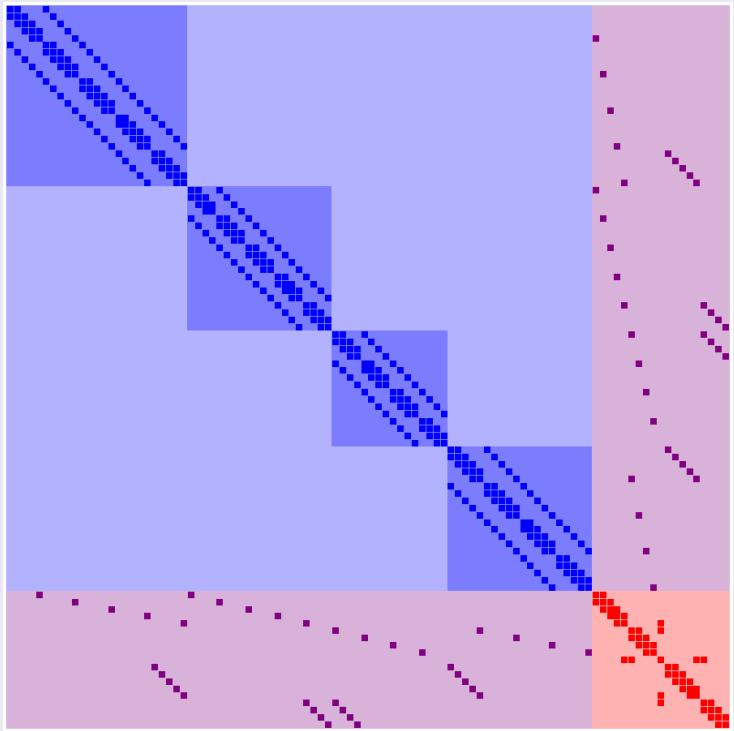


MaPHyS/Local Components

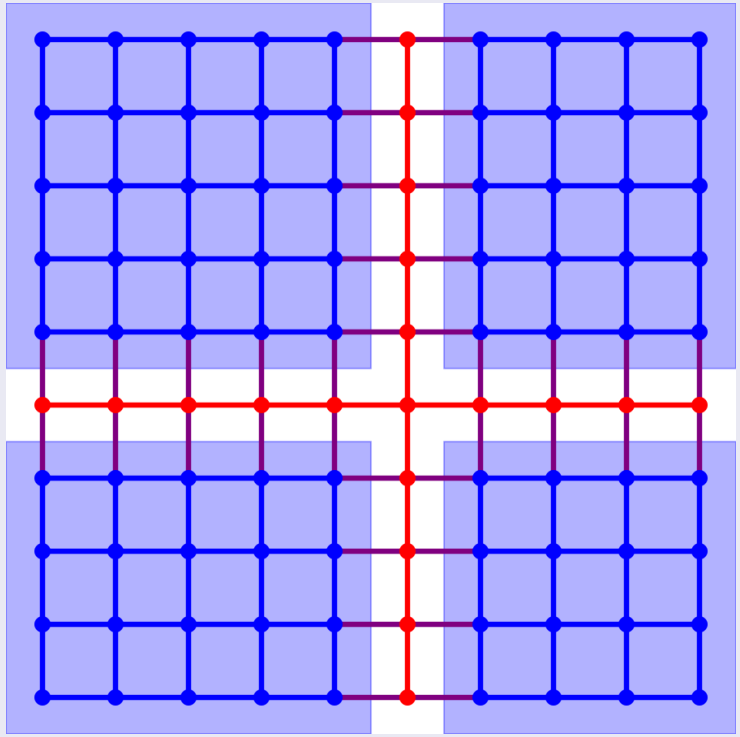
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A



Adjacency graph G

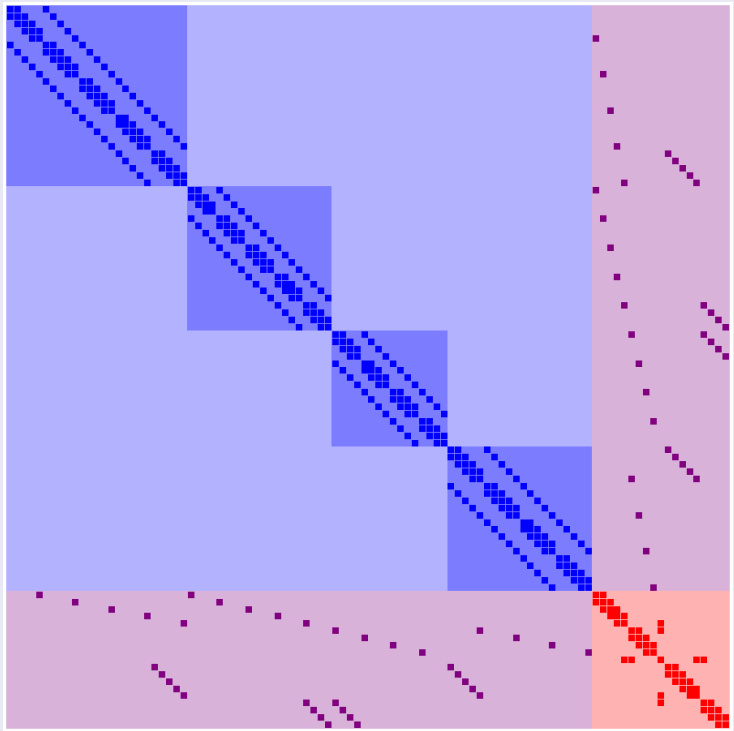


MaPHyS/Local Components

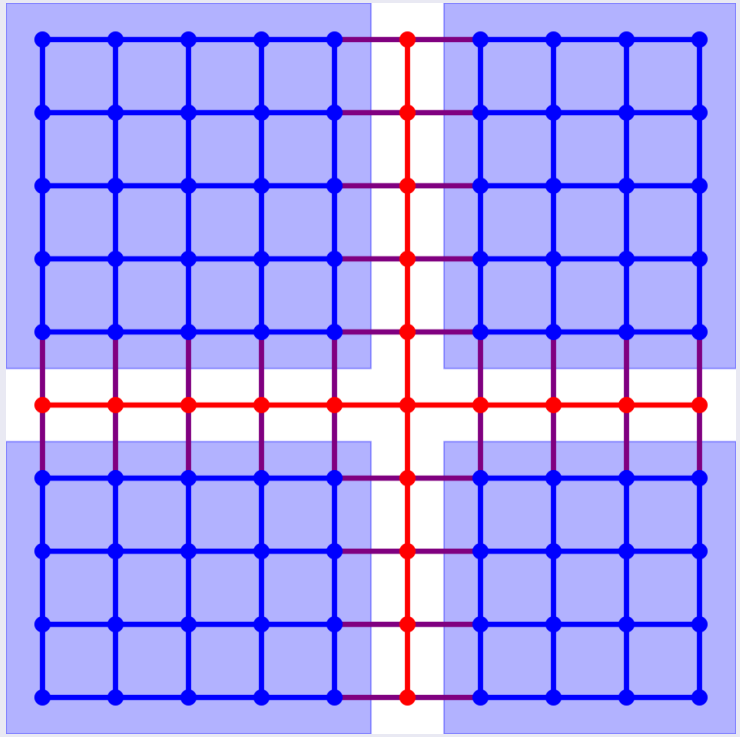
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A



Adjacency graph G

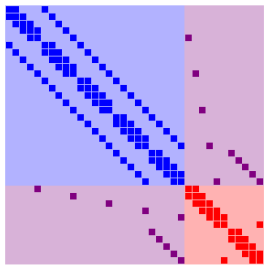


MaPHyS/Local Components

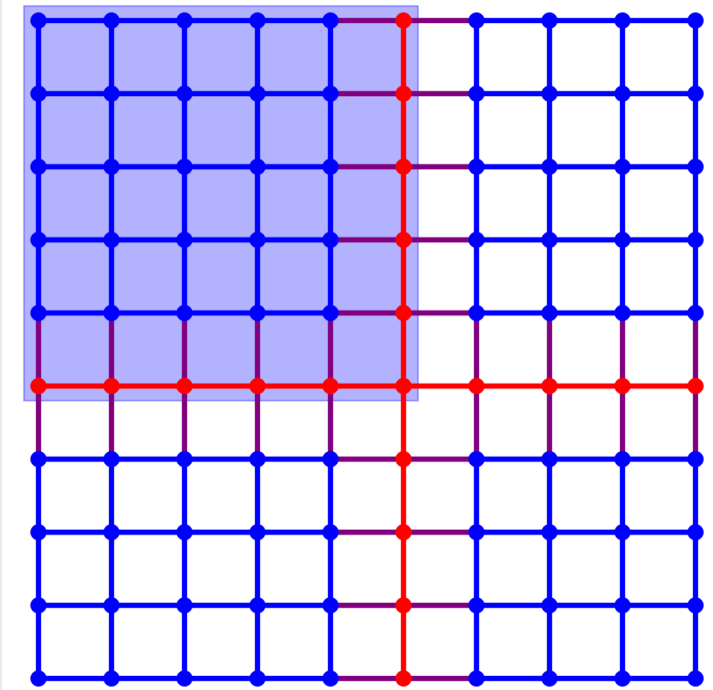
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A_i



Adjacency graph G

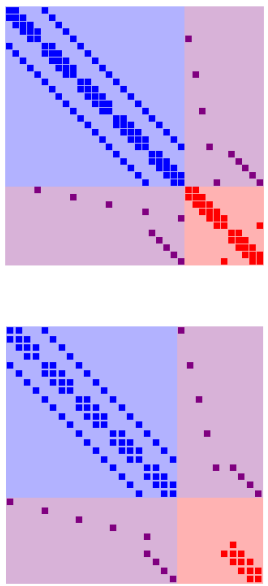


MaPHyS/Local Components

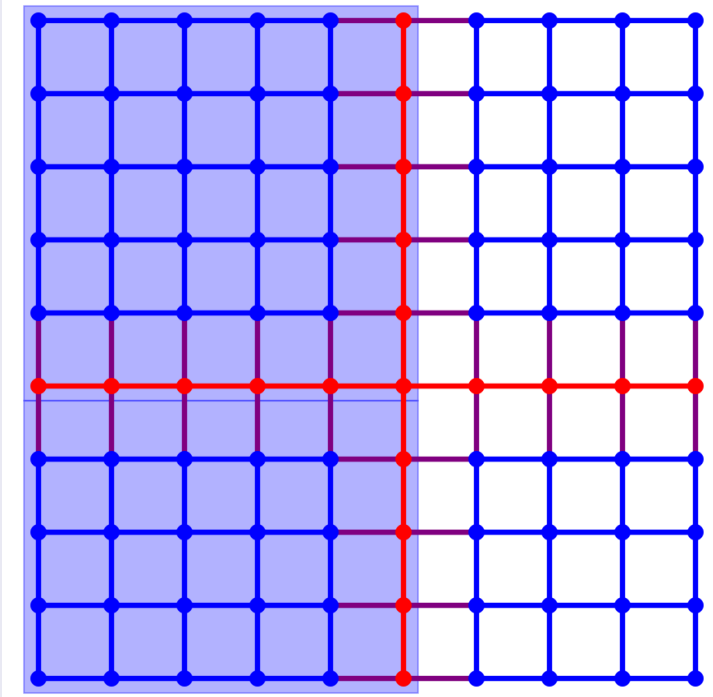
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A_i



Adjacency graph G

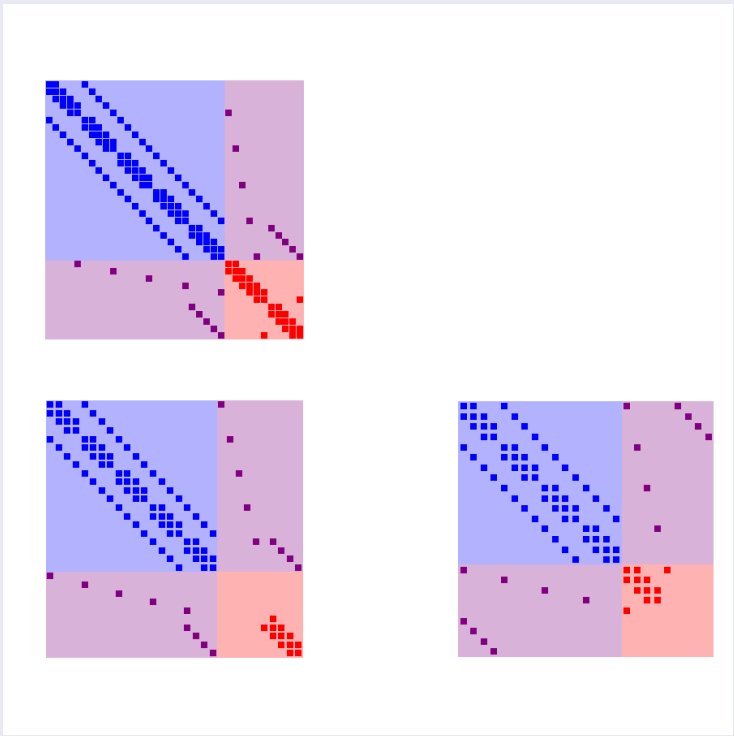


MaPHyS/Local Components

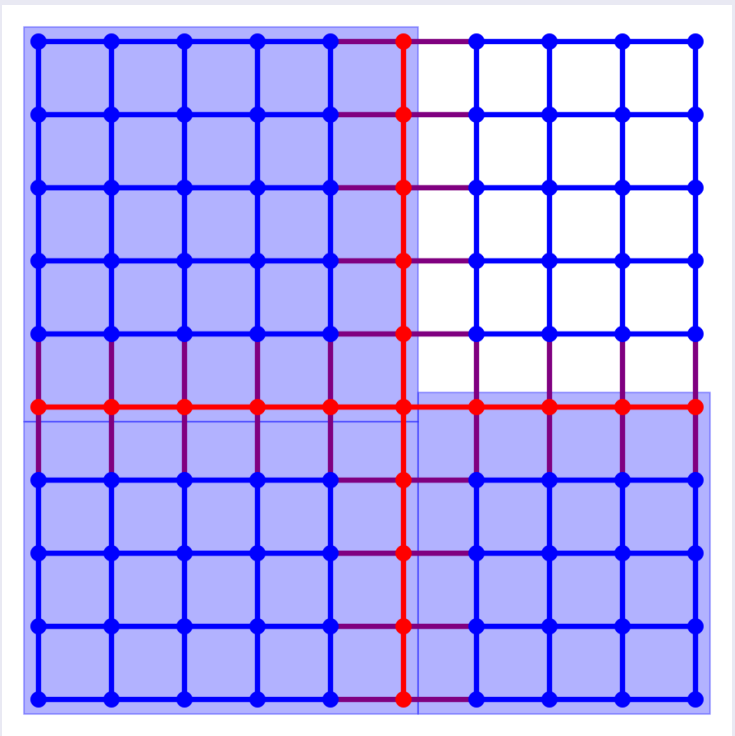
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A_i



Adjacency graph G

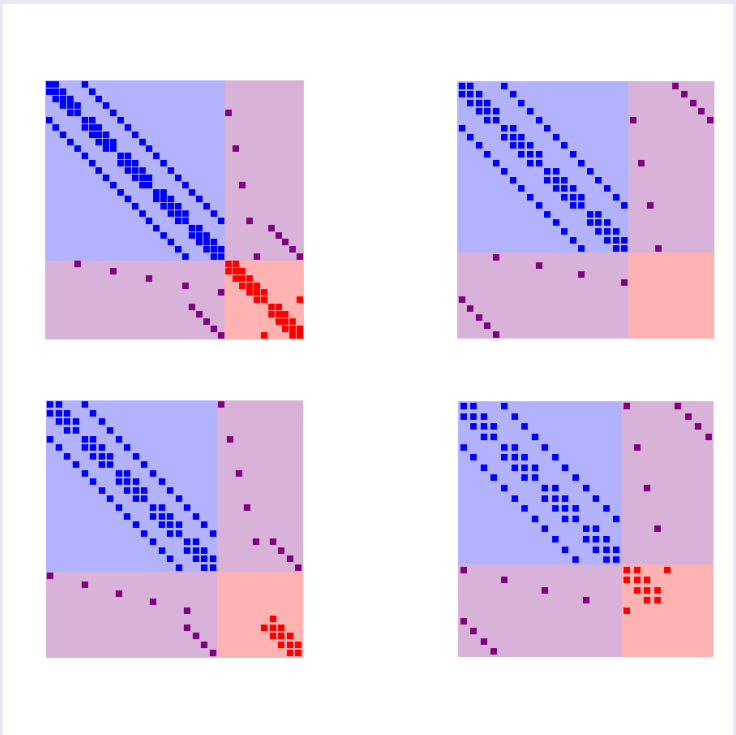


MaPHyS/Local Components

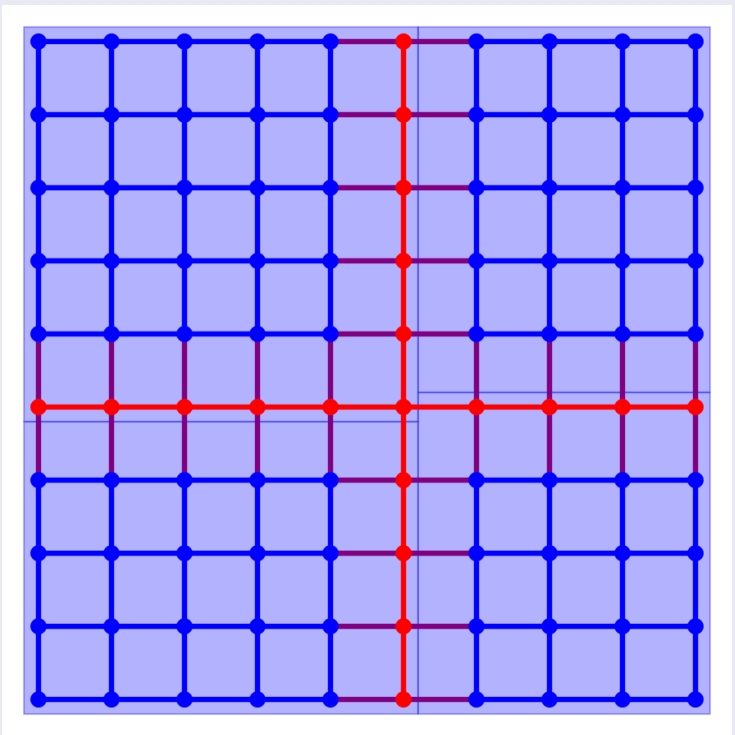
Local Schur complements are derived via the domain decomp.

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A^w_{\Gamma_i \Gamma_i} \end{pmatrix} \Rightarrow S_i = A^w_{\Gamma_i \Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

Matrix A_i



Adjacency graph G



MaPHyS/Step by Step

Analysis

- Graph partitioning (SCOTCH,METIS)

Factorization

- Factorization of $A_{I_i I_i}$ (PaStiX,MUMPS)
  (for applying: $A_{\Gamma_i I_i} A_{I_i I_i}^{-1} A_{I_i \Gamma_i}$)

Preconditioning

- $M^{AS} \approx S^{-1}$ (To be discussed later)

Solve

- on Γ: $M^{AS} S\, x_\Gamma = M^{AS}\, f$ (CERFACS Krylov package)
- on I: $x_I = A_{II}^{-1} (b_I - A_{I\Gamma} x_\Gamma)$

Preconditioning/Additive Schwarz Method

Motivation

- Schur system $\sim$ preconditioning.
- Further preconditioning for Krylov iterations.

Preconditioning/Additive Schwarz Method

Motivation

- Schur system $\sim$ preconditioning.
- Further preconditioning for Krylov iterations.

Additive Schwarz

- Additive Schwarz: algebraic and highly parallelizable.

$$M^{AS} = \sum_{i=1}^{N} R_i^T \, \overline{S}_i^{-1} R_i$$

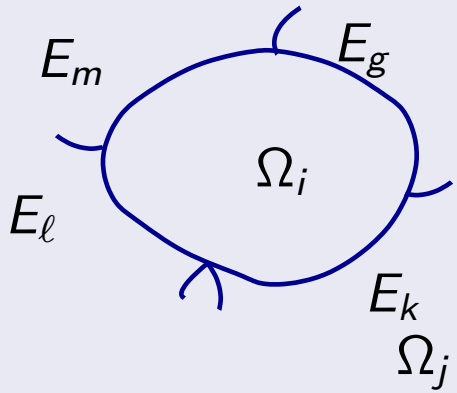
where $\overline{S}_i = R_i S R_i^T$ is a restriction of the Schur complement,

$$S = \sum_{i=1}^{N} R_i^T \, S_i \, R_i$$

Preconditioning/Additive Schwarz Method

$$S_i = A^w_{\Gamma_i\Gamma_i} - A_{\Gamma_i I_i} A^{-1}_{I_i I_i} A_{I_i \Gamma_i}$$

$$M^{AS} = \sum_{i=1}^{\#domains} R_i^T (\bar{S}_i)^{-1} R_i$$



$$\bar{S}_i = \begin{pmatrix} S_{mm} & S_{mg} & S_{mk} & S_{m\ell} \\ S_{gm} & S_{gg} & S_{gk} & S_{g\ell} \\ S_{km} & S_{kg} & S_{kk} & S_{k\ell} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S_{\ell\ell} \end{pmatrix} \qquad S_i = \begin{pmatrix} S^i_{mm} & S_{mg} & S_{mk} & S_{m\ell} \\ S_{gm} & S^i_{gg} & S_{gk} & S_{g\ell} \\ S_{km} & S_{kg} & S^i_{kk} & S_{k\ell} \\ S_{\ell m} & S_{\ell g} & S_{\ell k} & S^i_{\ell\ell} \end{pmatrix}$$

$$S_{mm} = \sum_{j \in adj(m)} S^j_{mm}$$

Preconditioning/Additive Schwarz Method

Remarks and conclusions

Preconditioning/Additive Schwarz Method

Remarks and conclusions

- Bottlenecks:
  - Forming and Storing the factorization of $\overline{S}_i$
  - CPU and memory expensive.

Preconditioning/Additive Schwarz Method

Remarks and conclusions

- Bottlenecks:
  - Forming and Storing the factorization of $\overline{S}_i$
  - CPU and memory expensive.
- Tricks:
  - Sparsification strategy through dropping

$$\widehat{s}_{k\ell} = \begin{cases} \bar{s}_{k\ell} & \text{if} \quad \bar{s}_{k\ell} \geq \xi(|\bar{s}_{kk}| + |\bar{s}_{\ell\ell}|) \\ 0 & \text{else} \end{cases}$$

  - Approximation through ILU
  - Difficulties: CPU time, tuning, memory.

Preconditioning/Additive Schwarz Method

Remarks and conclusions

- Bottlenecks:
  - Forming and Storing the factorization of $\overline{S}_i$
  - CPU and memory expensive.
- Tricks:
  - Sparsification strategy through dropping

$$\widehat{s}_{k\ell} = \begin{cases} \bar{s}_{k\ell} & \text{if} \quad \bar{s}_{k\ell} \geq \xi(|\bar{s}_{kk}| + |\bar{s}_{\ell\ell}|) \\ 0 & \text{else} \end{cases}$$

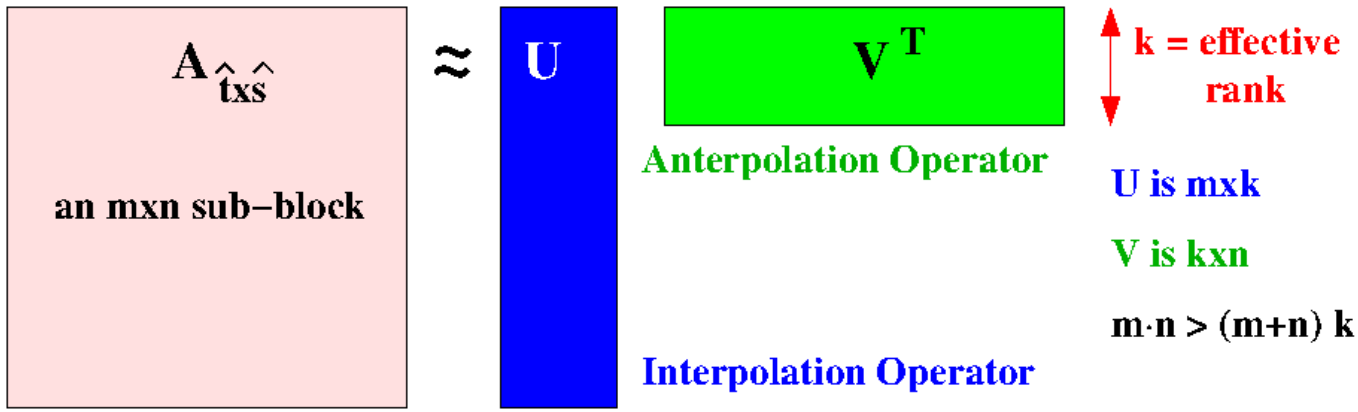
  - Approximation through ILU
  - Difficulties: CPU time, tuning, memory.
- H-matrix techniques are expected to resolve these issues.

H-matrices/Informal Definitions

- **Hierarchical matrix** is a data sparse approximation of a non-sparse matrix.

H-matrices/Informal Definitions

- **Hierarchical matrix** is a data sparse approximation of a non-sparse matrix.
- **Basic principles**
  1. perform rows and columns permutations
  2. replace sub-blocks by low-rank factorizations.

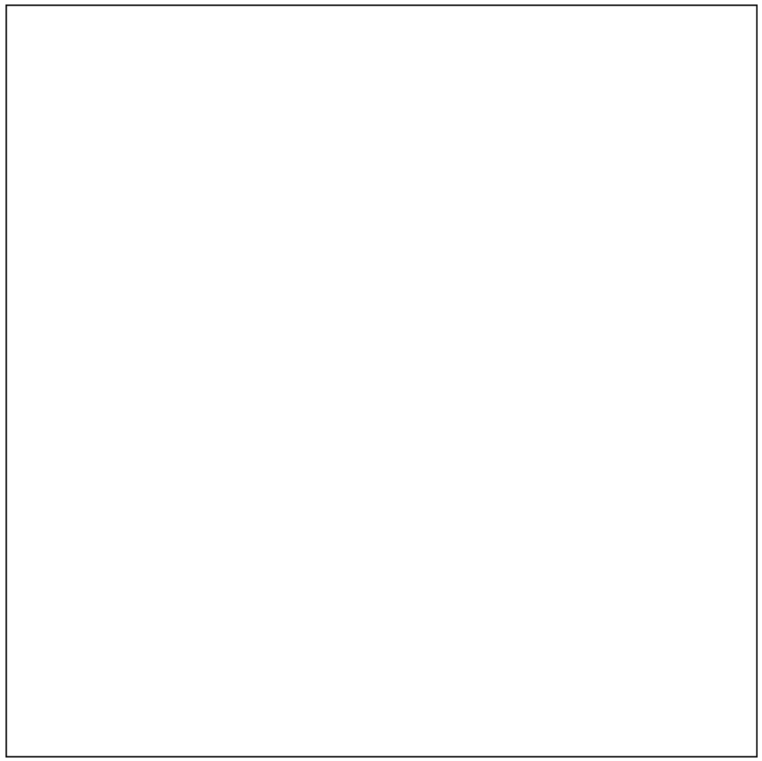


$A_{\hat{t}\times\hat{s}}$ is a sub-block of $A \in \mathbb{F}^{N\times N}$, $\hat{s}, \hat{t} \subset \mathcal{I} = \{1, 2, ..., N\}$.

3. **Hierarchical partitioning** $\Rightarrow$ almost linear complexity.
   $\Leftrightarrow \mathcal{O}\left(N \log^{\alpha} N\right)$, α is 'small'

H-matrices/Hierarchical Partitioning

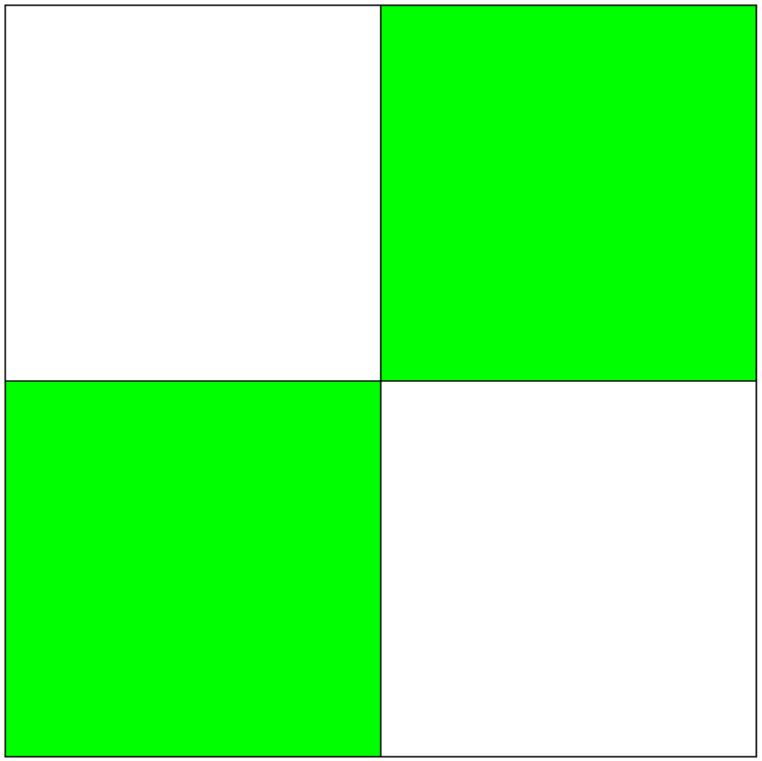
Strong Hierarchy:



Start Level:
Input Matrix

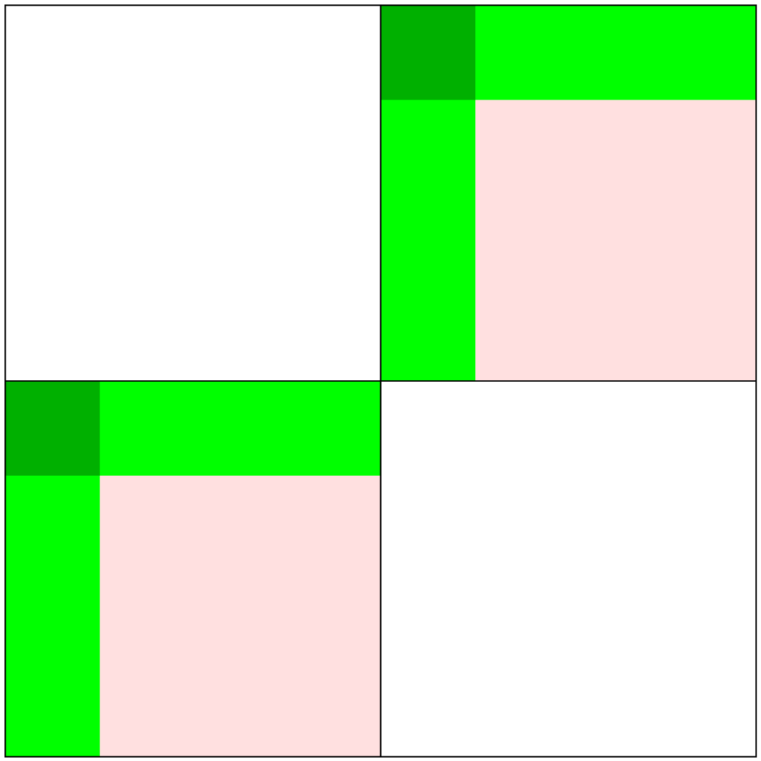
H-matrices/Hierarchical Partitioning

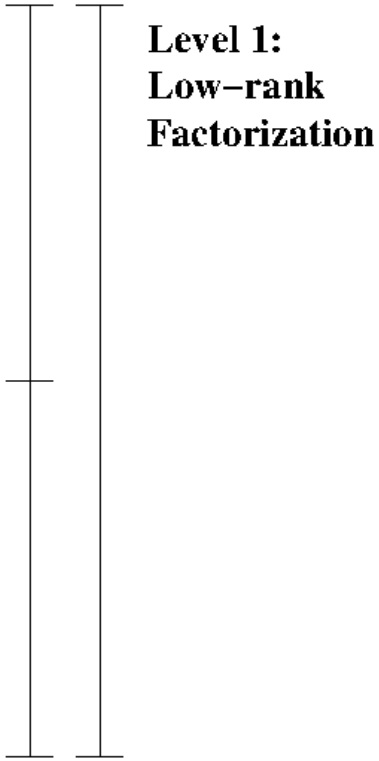
Strong Hierarchy:



H-matrices/Hierarchical Partitioning

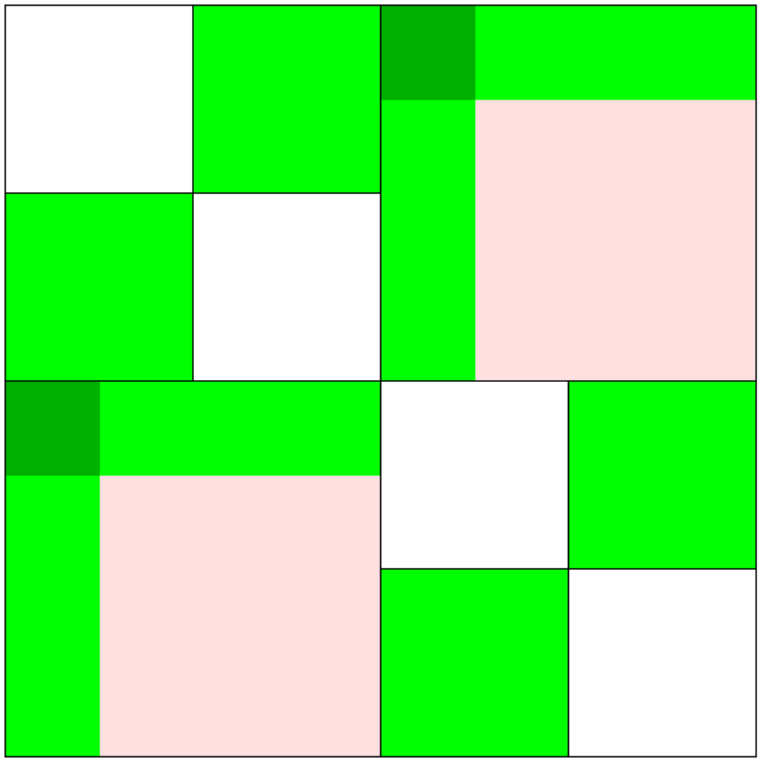
Strong Hierarchy:

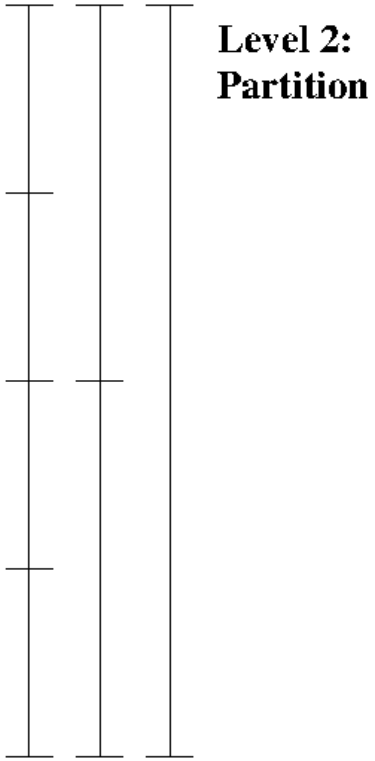




H-matrices/Hierarchical Partitioning

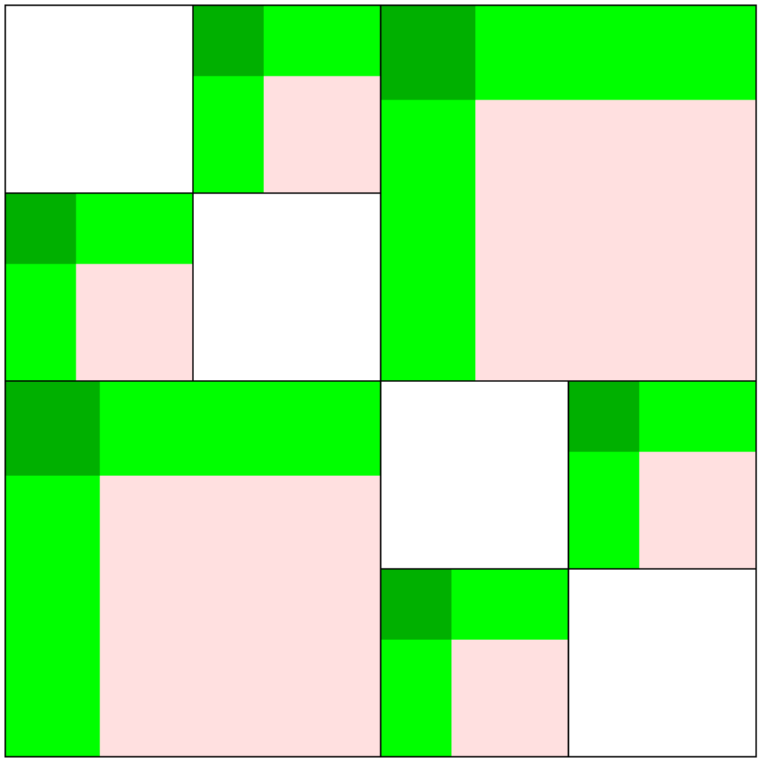
Strong Hierarchy:

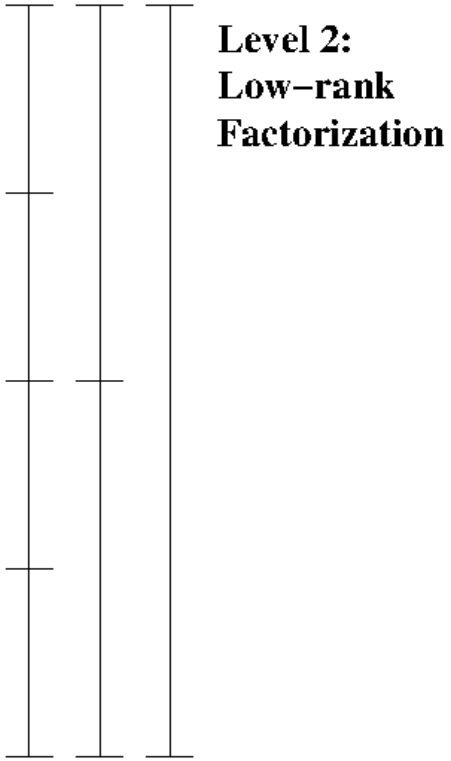




H-matrices/Hierarchical Partitioning

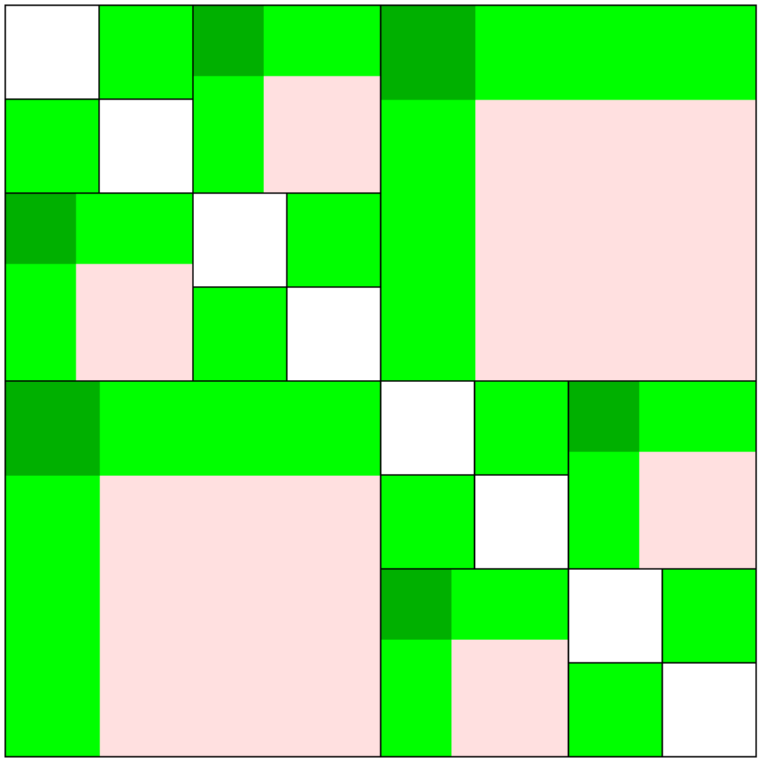
Strong Hierarchy:

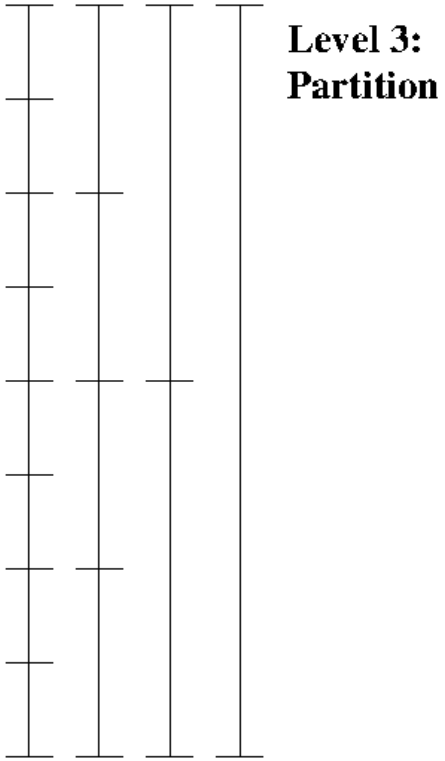




H-matrices/Hierarchical Partitioning

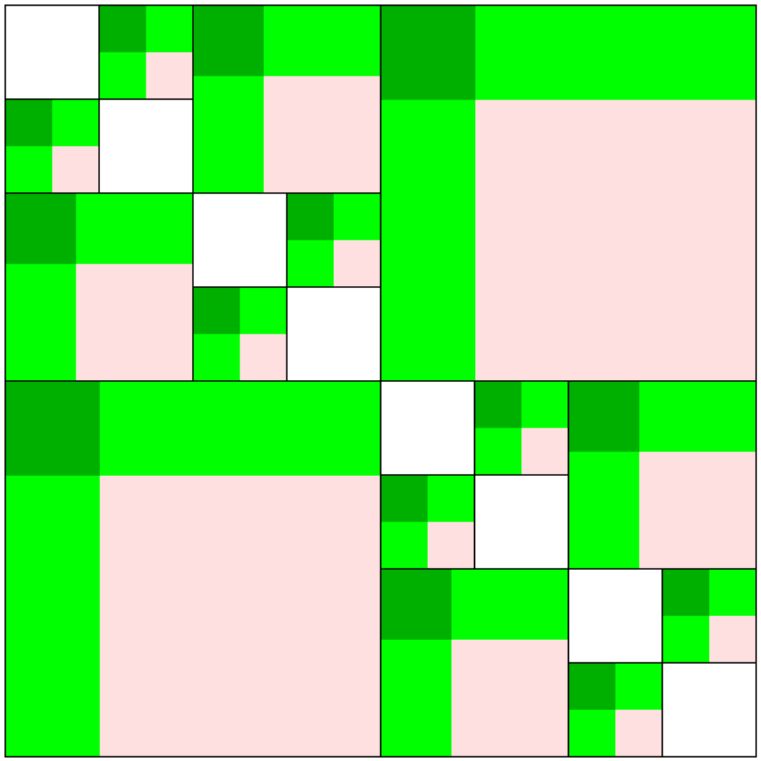
Strong Hierarchy:

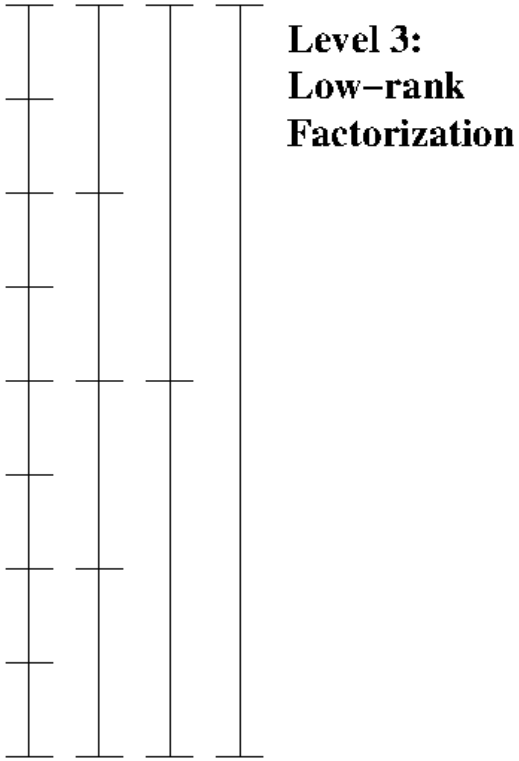




H-matrices/Hierarchical Partitioning

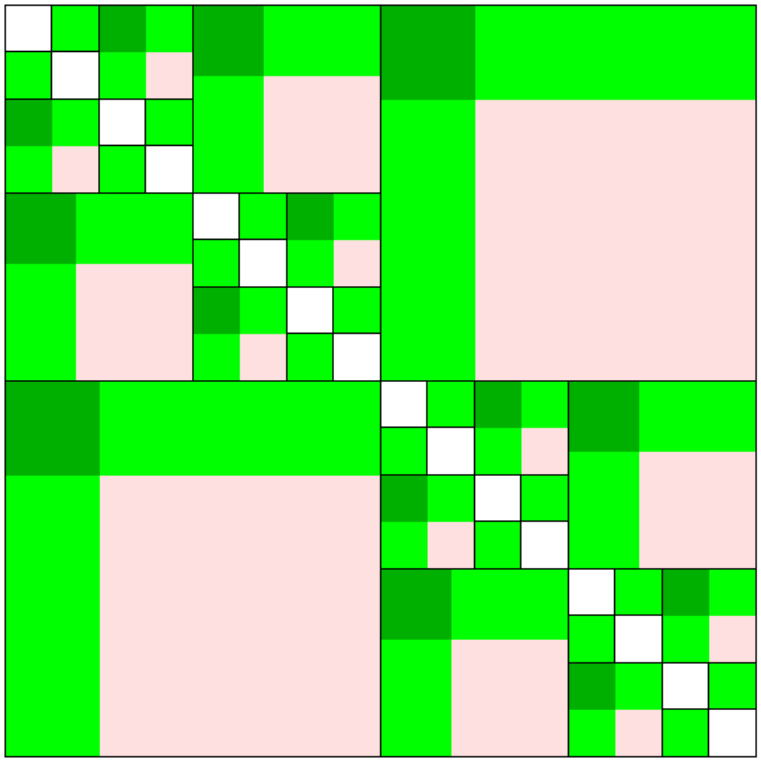
Strong Hierarchy:

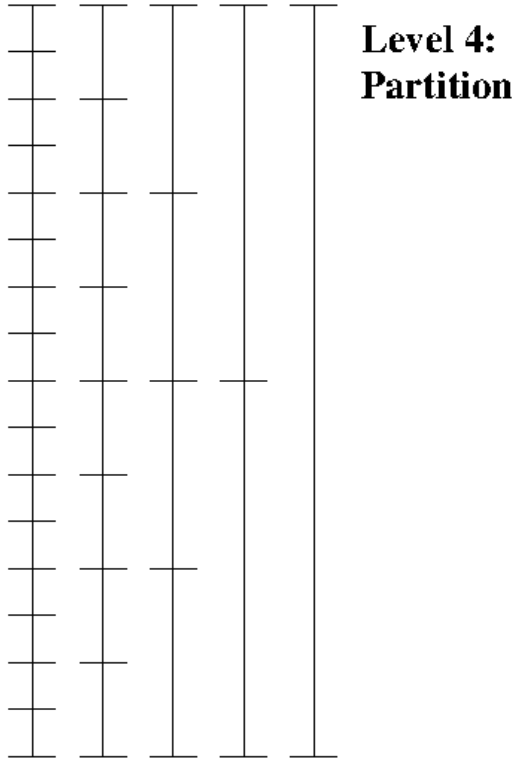




H-matrices/Hierarchical Partitioning

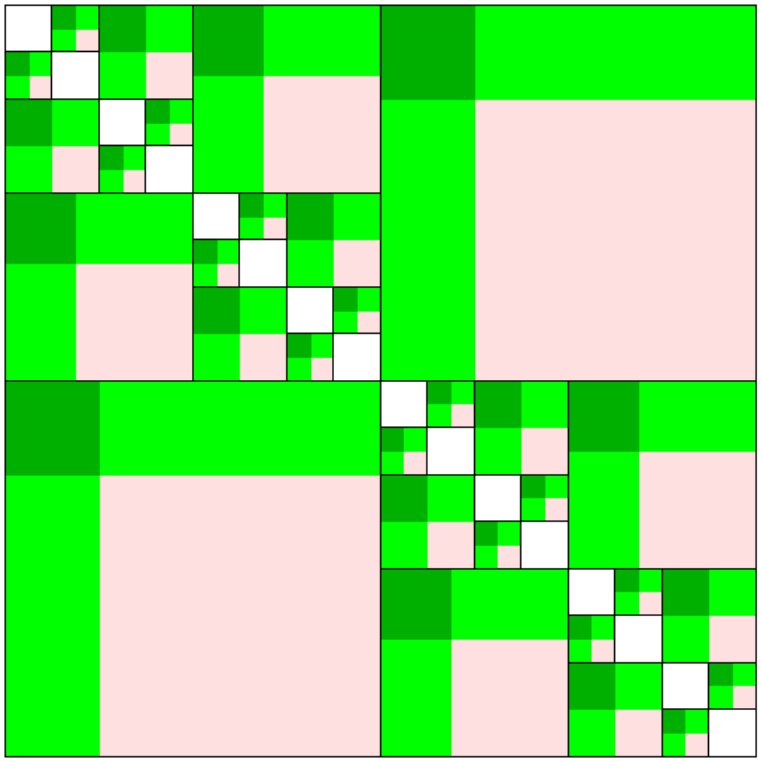
Strong Hierarchy:

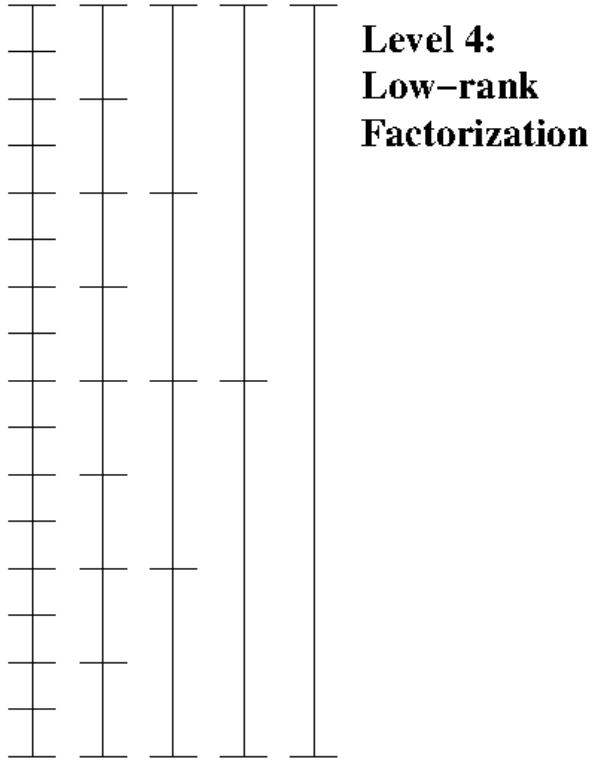




H-matrices/Hierarchical Partitioning

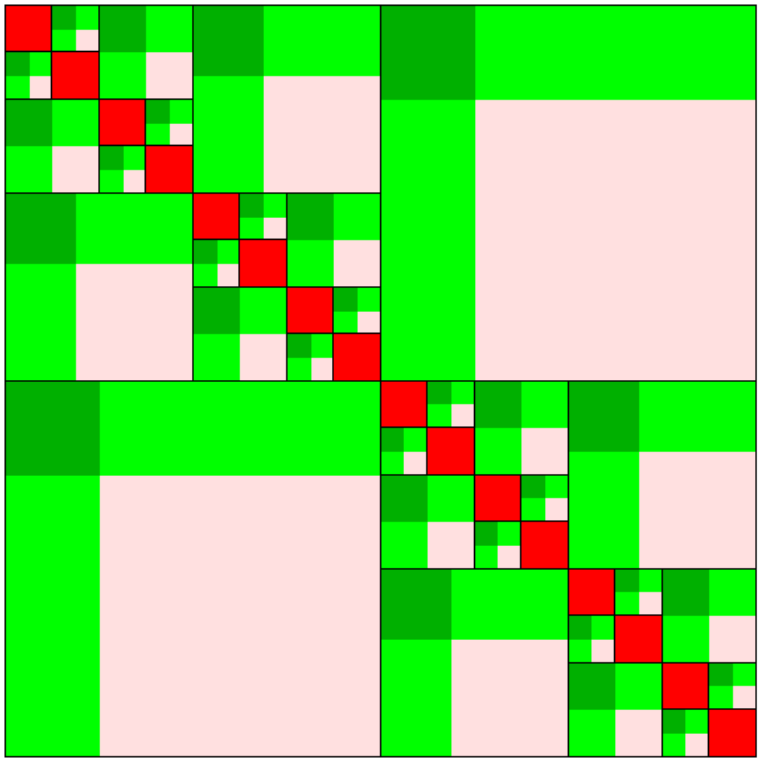
Strong Hierarchy:

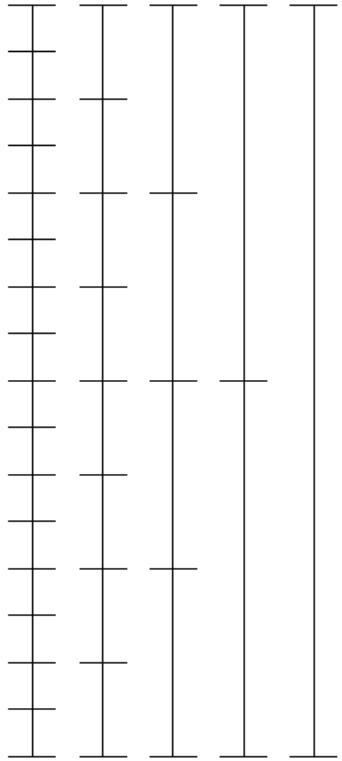




H-matrices/Hierarchical Partitioning

Strong Hierarchy:

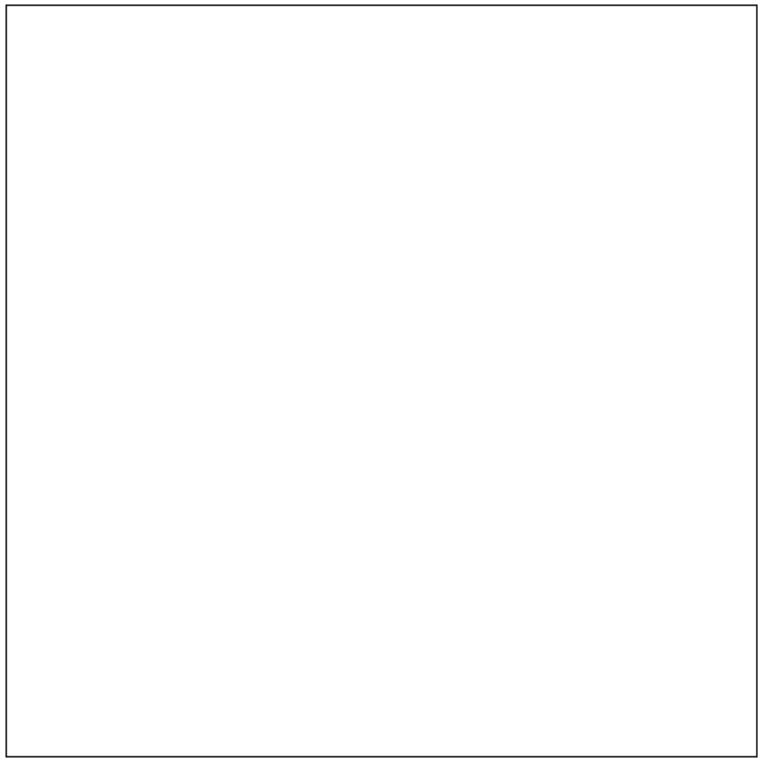




Level 5:
Stop:
remaining blocks
are small enough

H-matrices/Hierarchical Partitioning

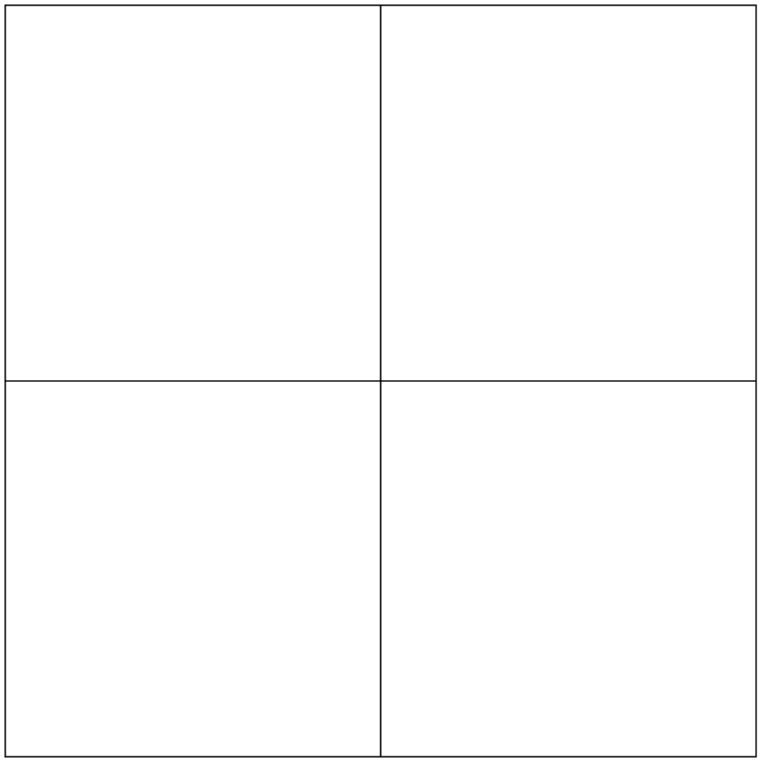
Weak Hierarchy:



Start Level:
Input Matrix

H-matrices/Hierarchical Partitioning

Weak Hierarchy:

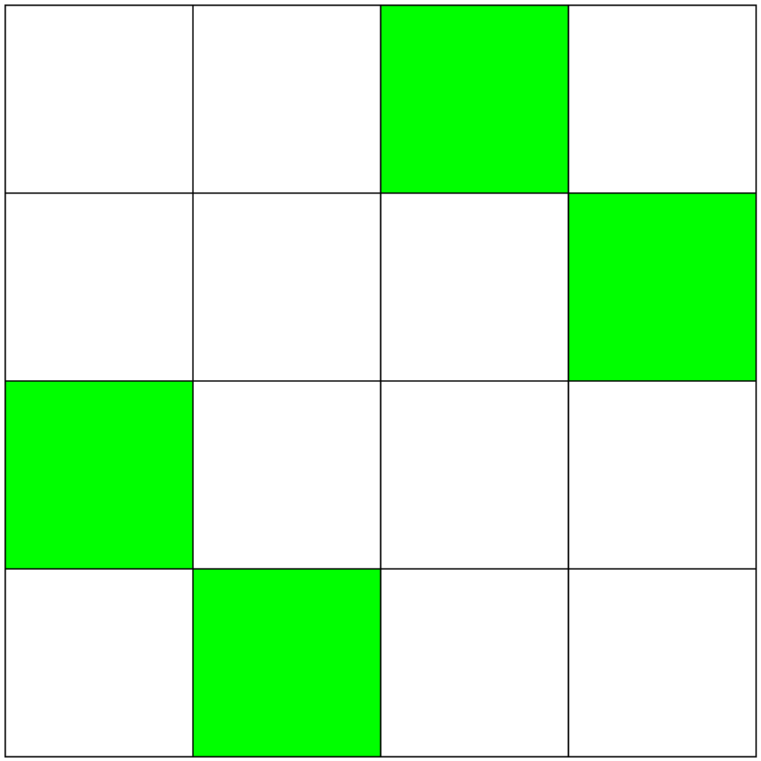


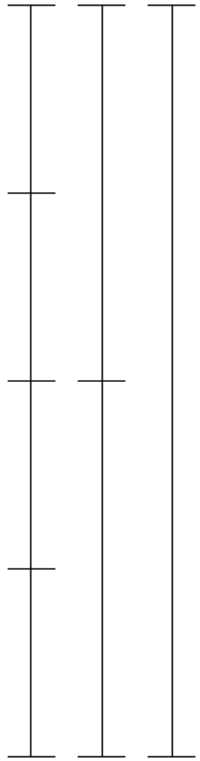
Level 1:
Partition

Off−diagonal blocks are not low−rank

H-matrices/Hierarchical Partitioning

Weak Hierarchy:



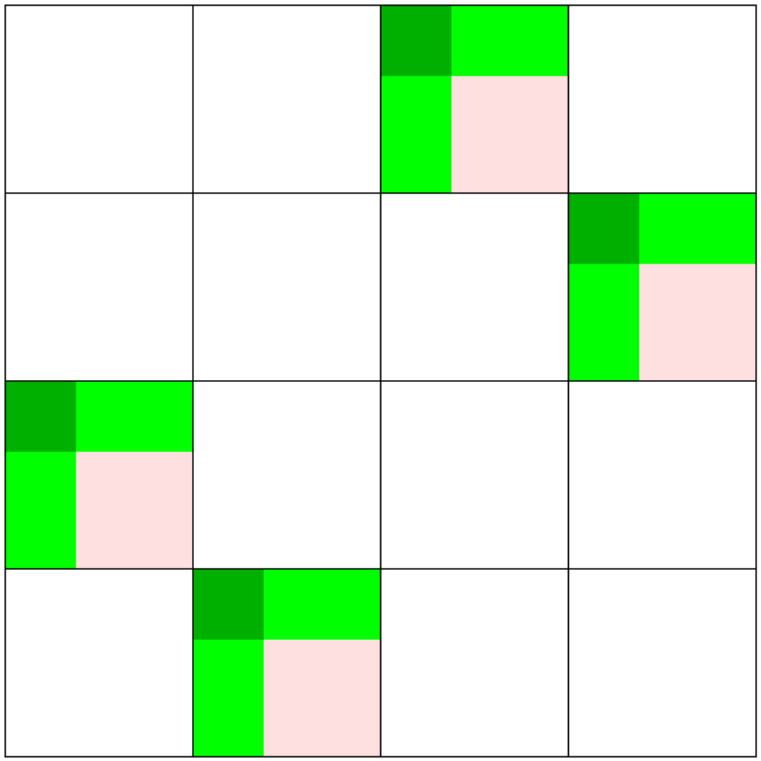


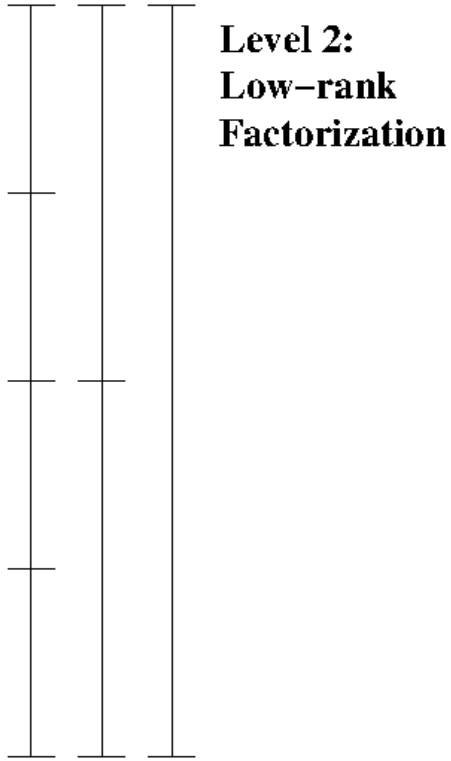
Level 2: Partition

Partition off–diagonal blocks of Level 1

H-matrices/Hierarchical Partitioning

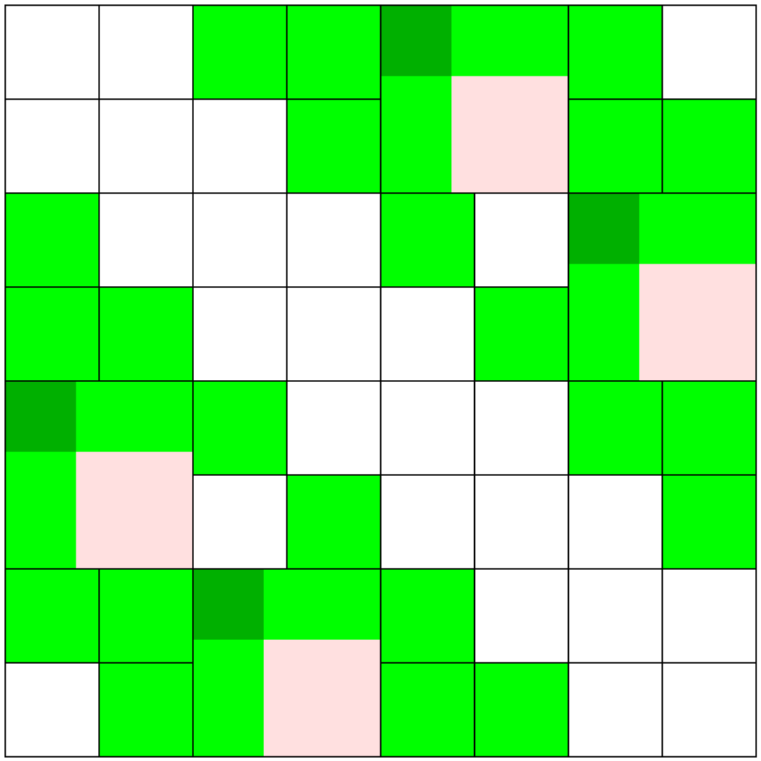
Weak Hierarchy:

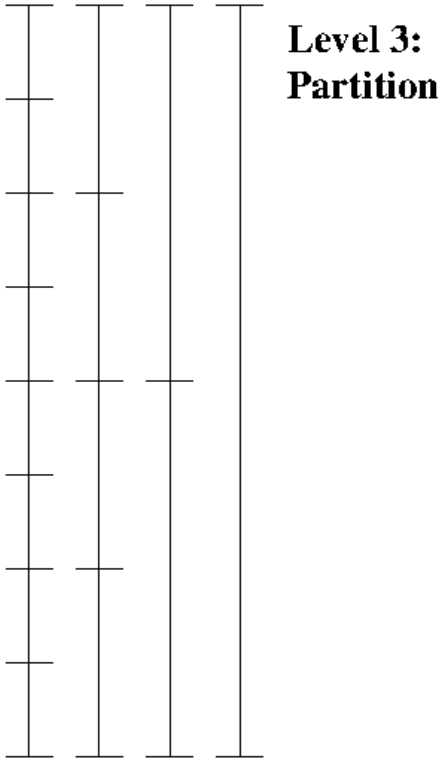




H-matrices/Hierarchical Partitioning

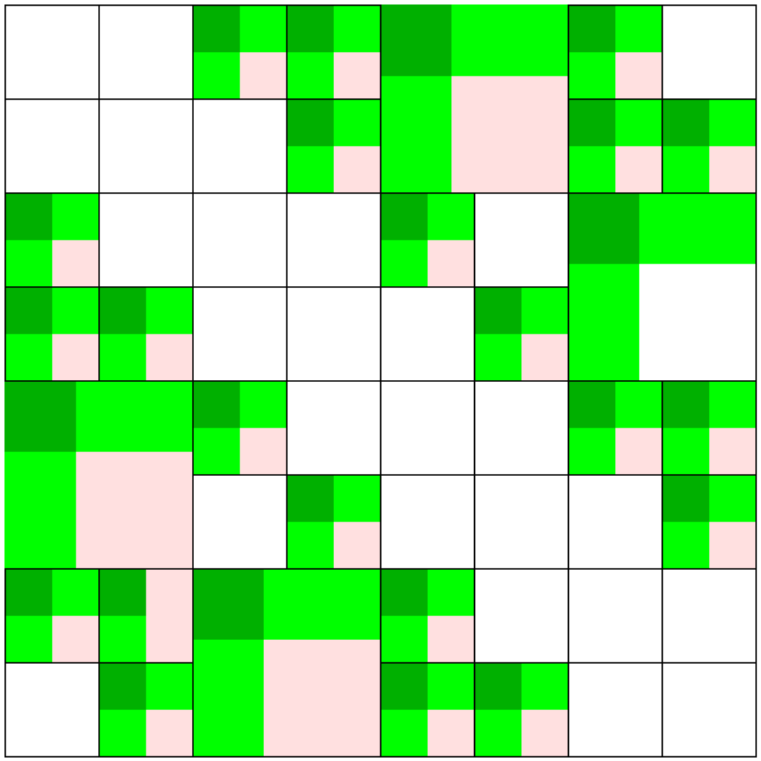
Weak Hierarchy:

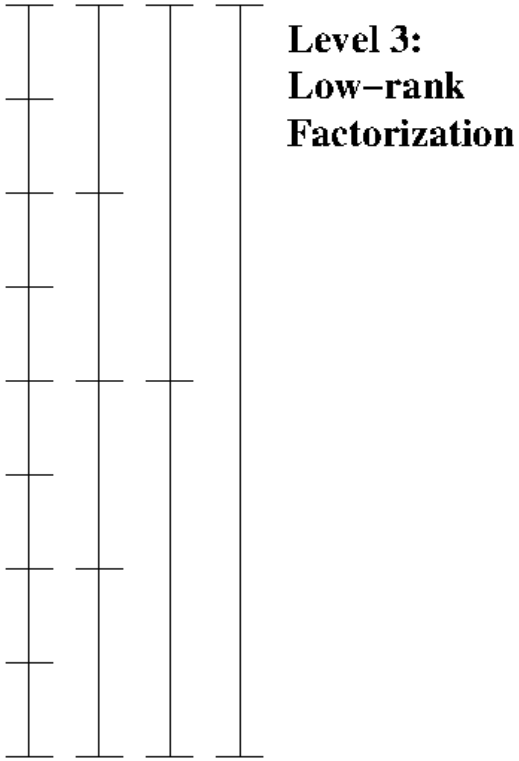




H-matrices/Hierarchical Partitioning

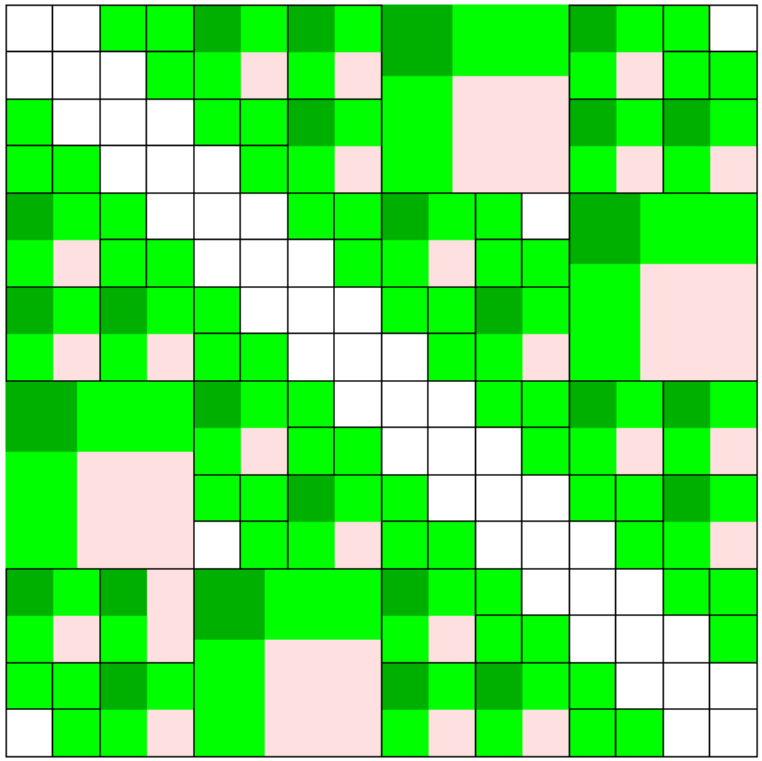
Weak Hierarchy:

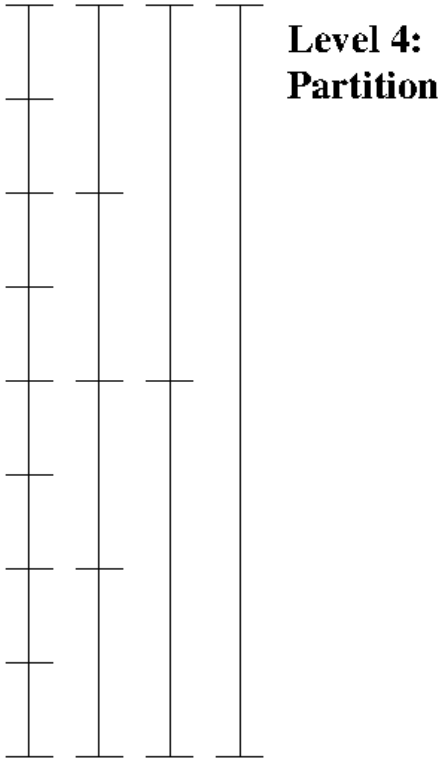




H-matrices/Hierarchical Partitioning

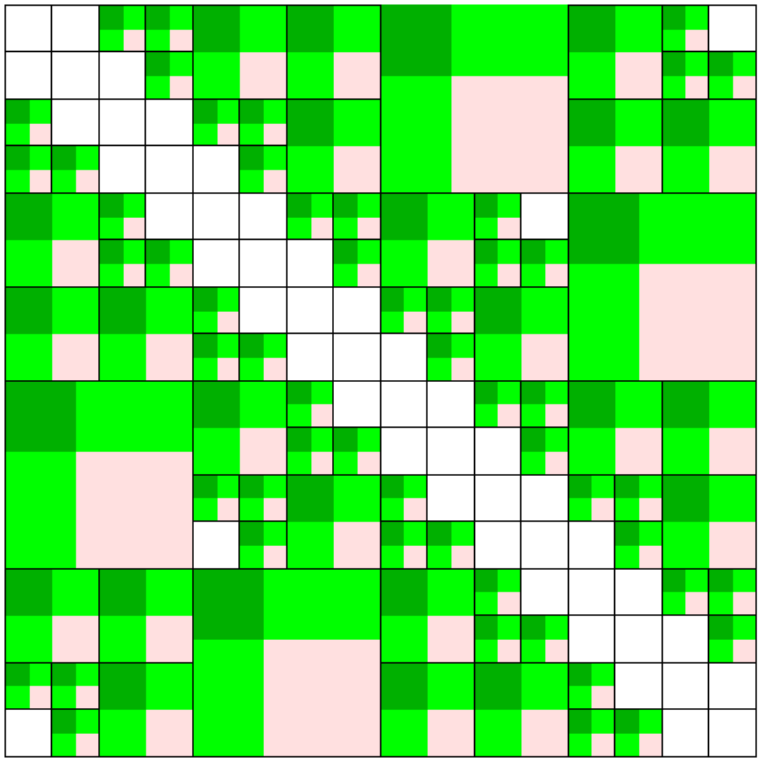
Weak Hierarchy:

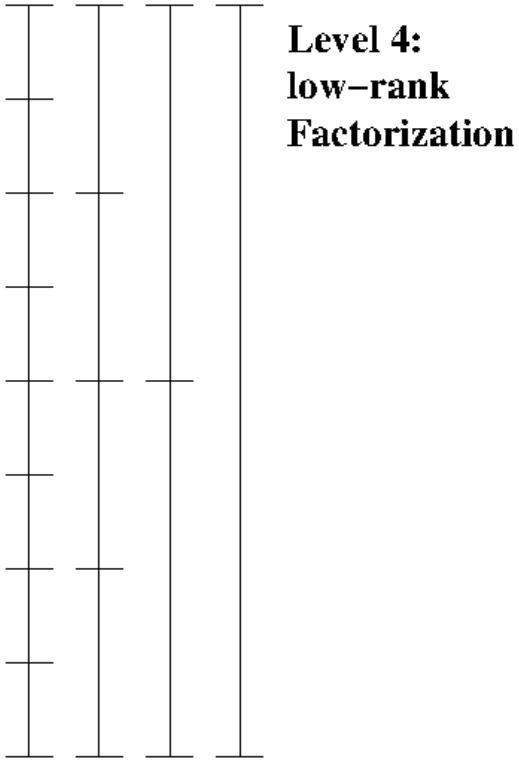




H-matrices/Hierarchical Partitioning

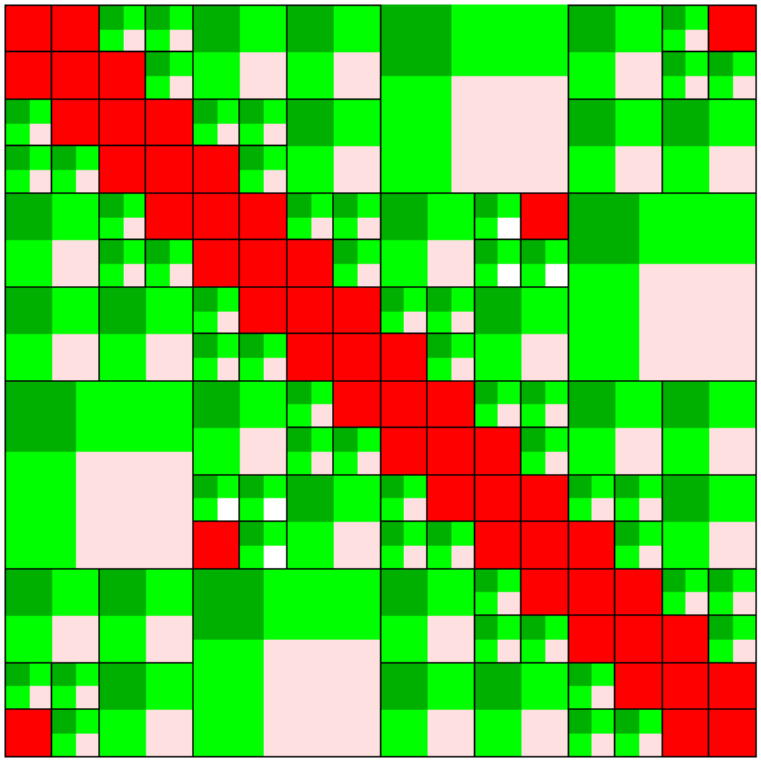
Weak Hierarchy:

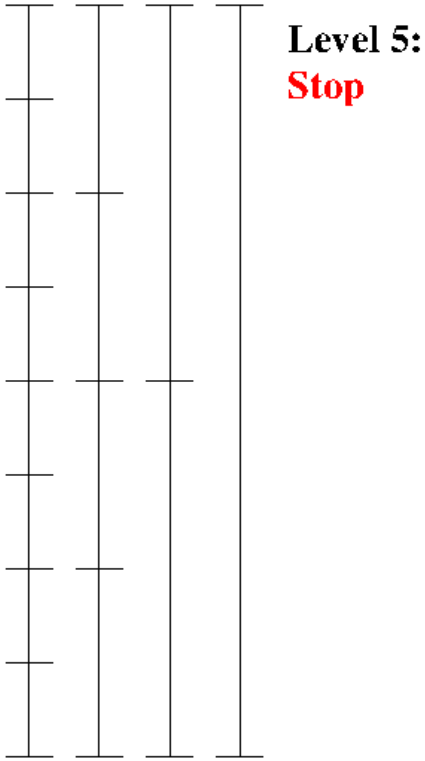




H-matrices/Hierarchical Partitioning

Weak Hierarchy:





H-matrices/ MaPHyS

Implementation within MaPHyS

- Dense H-matrix solver.
- Non nested strong hierarchy: **HODLR**.
- Low-rank blocks identification via graph partitioning.
- Low-rank factorization: SVD, Random, ACA, BDLR.

H-matrices/ MaPHyS

Implementation within MaPHyS

- Dense H-matrix solver.
- Non nested strong hierarchy: **HODLR**.
- Low-rank blocks identification via graph partitioning.
- Low-rank factorization: SVD, Random, ACA, BDLR.

Remarks

- Compression & operations can be improved with nested bases.
- **HODLR** is sufficient for testing the preconditioning.

H-matrices/ MaPHyS

Implementation within MaPHyS

- Dense H-matrix solver.
- Non nested strong hierarchy: **HODLR**.
- Low-rank blocks identification via graph partitioning.
- Low-rank factorization: SVD, Random, ACA, BDLR.

Remarks

- Compression & operations can be improved with nested bases.
- **HODLR** is sufficient for testing the preconditioning.

The inversion algorithm

A Fast Block Low-Rank Dense Solver with Applications to FE Matrices, *A. Aminfar , E. Darve , S. Ambikasaran, Stanford,* **2014**.

H-matrices/Fast HODLR Inversion

Partitioned system

$$Kx = \begin{pmatrix} K_1 & M_{1,2} \\ M_{2,1} & K_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = F \in \mathbb{F}^{(n_1+n_2)\times nrhs},$$

$K_1 \in \mathbb{F}^{n_1 \times n_1}$, $K_2 \in \mathbb{F}^{n_2 \times n_2}$.

H-matrices/Fast HODLR Inversion

Partitioned system

$$Kx = \begin{pmatrix} K_1 & M_{1,2} \\ M_{2,1} & K_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = F \in \mathbb{F}^{(n_1+n_2)\times nrhs},$$

$K_1 \in \mathbb{F}^{n_1 \times n_1}$, $K_2 \in \mathbb{F}^{n_2 \times n_2}$.

Equivalent system

$$\begin{aligned} x_1 &= \underbrace{K_1 \backslash F_1}_{c_1 \in \mathbb{F}^{n_1 \times nrhs}} - \underbrace{(K_1 \backslash M_{1,2})}_{b_1 \in \mathbb{F}^{n_1 \times n_2}} x_2 , \\ x_2 &= \underbrace{K_2 \backslash F_2}_{c_2 \in \mathbb{F}^{n_2 \times nrhs}} - \underbrace{(K_2 \backslash M_{2,1})}_{b_2 \in \mathbb{F}^{n_2 \times n_1}} x_1 . \end{aligned}$$

H-matrices/Fast HODLR Inversion

Partitioned system

$$Kx = \begin{pmatrix} K_1 & M_{1,2} \\ M_{2,1} & K_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = F \in \mathbb{F}^{(n_1+n_2)\times nrhs},$$

$K_1 \in \mathbb{F}^{n_1 \times n_1}$, $K_2 \in \mathbb{F}^{n_2 \times n_2}$.

Equivalent system - low rank off-diagonal blocks

$$\begin{aligned} x_1 &= \underbrace{K_1 \backslash F_1}_{c_1 \in \mathbb{F}^{n_1 \times nrhs}} - \underbrace{(K_1 \backslash U_1)}_{d_1 \in \mathbb{F}^{n_1 \times k_1}} \cdot \underbrace{V_{1,2}^T x_2}_{y_2 \in \mathbb{F}^{k_1 \times nrhs}}, \\ x_2 &= \underbrace{K_2 \backslash F_2}_{c_2 \in \mathbb{F}^{n_2 \times nrhs}} - \underbrace{(K_2 \backslash U_2)}_{d_2 \in \mathbb{F}^{n_2 \times k_2}} \cdot \underbrace{V_{2,1}^T x_1}_{y_1 \in \mathbb{F}^{k_2 \times nrhs}}. \end{aligned}$$

H-matrices/Fast HODLR Inversion

Partitioned system

$$Kx = \begin{pmatrix} K_1 & M_{1,2} \\ M_{2,1} & K_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = F \in \mathbb{F}^{(n_1+n_2)\times nrhs},$$

$K_1 \in \mathbb{F}^{n_1 \times n_1}$, $K_2 \in \mathbb{F}^{n_2 \times n_2}$.

Reduced system - low rank off-diagonal blocks

$$\begin{aligned} y_1 &= \underbrace{(V_{2,1}^T \cdot c_1)}_{k_2 \times nrhs} - \underbrace{(V_{2,1}^T \cdot d_1)}_{k_2 \times k_1} \cdot \underbrace{y_2}_{k_1 \times nrhs}, \\ y_2 &= \underbrace{(V_{1,2}^T \cdot c_2)}_{k_1 \times nrhs} - \underbrace{(V_{1,2}^T \cdot d_2)}_{k_1 \times k_2} \cdot \underbrace{y_1}_{k_2 \times nrhs}, \end{aligned}$$

Now apply recursively to K_i (HODLR).

Results/Cubes Model

Physical Setting

- Equations:

$$
\begin{array}{lrcl}
(1) & u_{xx} + u_{yy} + u_{zz} & = & f \\
(2) & 10u_{xx} + u_{yy} + u_{zz} & = & f \\
(3) & 100u_{xx} + u_{yy} + u_{zz} & = & f \\
(4) & 1000u_{xx} + u_{yy} + u_{zz} & = & f
\end{array}
$$

- $\Omega = [0,1]^3 \subset \mathbb{R}^3$ + Dirichlet B.C.

Results/Cubes Model

Physical Setting

- Equations:

$$\begin{array}{lll} (1) & u_{xx} + u_{yy} + u_{zz} & = f \\ (2) & 10u_{xx} + u_{yy} + u_{zz} & = f \\ (3) & 100u_{xx} + u_{yy} + u_{zz} & = f \\ (4) & 1000u_{xx} + u_{yy} + u_{zz} & = f \end{array}$$

- $\Omega = [0,1]^3 \subset \mathbb{R}^3$ + Dirichlet B.C.

Computational Setting

- 2^{nd} order FD 7-points stencil.
- $3^3 = 27$ subdomains - each handled by one *processor*.
- Geometry based domain decomposition.
- Stop criteria: $\frac{\|Sx_\Gamma - f\|}{\|b\|} = \frac{\|Ax-b\|}{\|b\|} < 1E-10$

Results/Cubes Model

Additive Schwarz H-matrix preconditioning

Results/Cubes Model

Additive Schwarz H-matrix preconditioning

- H-matrix construction:
  - Cluster tree via graph bisections.
  - Each off-diagonal block is approximated with SVD

$$\left\| M - UV^T \right\|_{2,op} \leq \epsilon \quad (\epsilon \text{ is predetermined})$$

Results/Cubes Model

Additive Schwarz H-matrix preconditioning

- H-matrix construction:
  - Cluster tree via graph bisections.
  - Each off-diagonal block is approximated with SVD

$$\left\| M - UV^T \right\|_{2,op} \leq \epsilon \quad (\epsilon \text{ is predetermined})$$

- iterations to convergence (GMRES) vs. compression ratio.

Results/Cubes Model

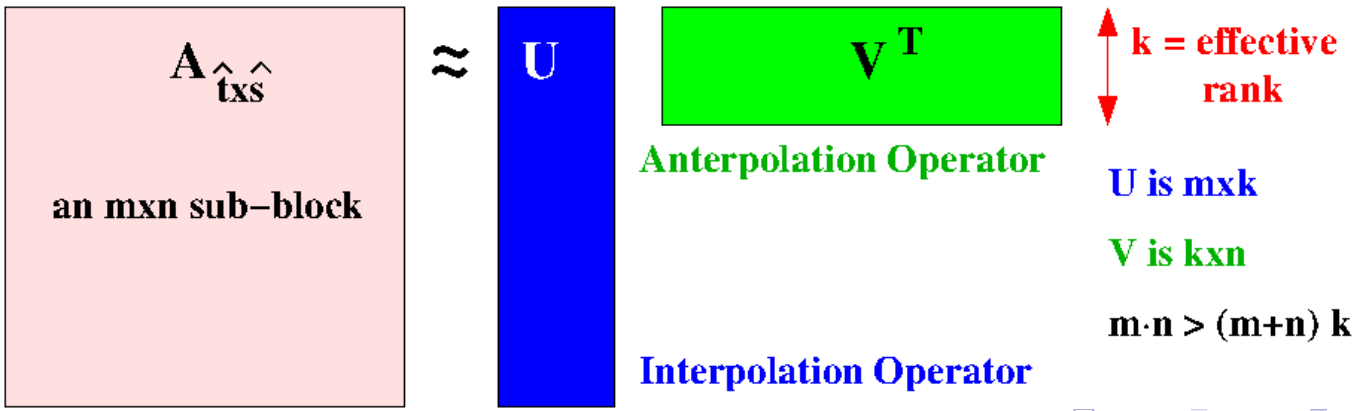
Additive Schwarz H-matrix preconditioning

- H-matrix construction:
  - Cluster tree via graph bisections.
  - Each off-diagonal block is approximated with SVD

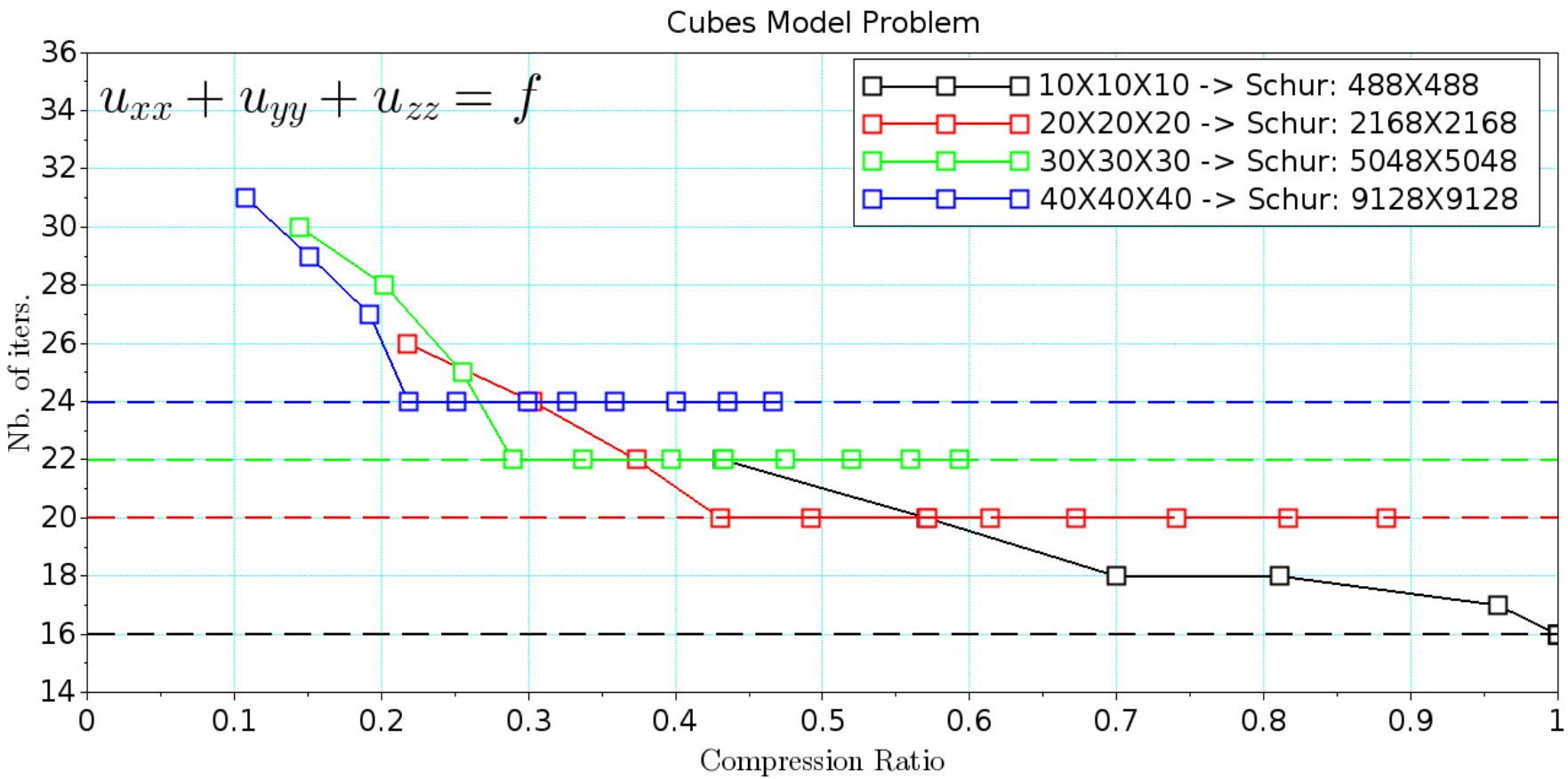
$$\left\| M - UV^T \right\|_{2,op} \leq \epsilon \quad (\epsilon \text{ is predetermined})$$

- iterations to convergence (GMRES) vs. compression ratio.

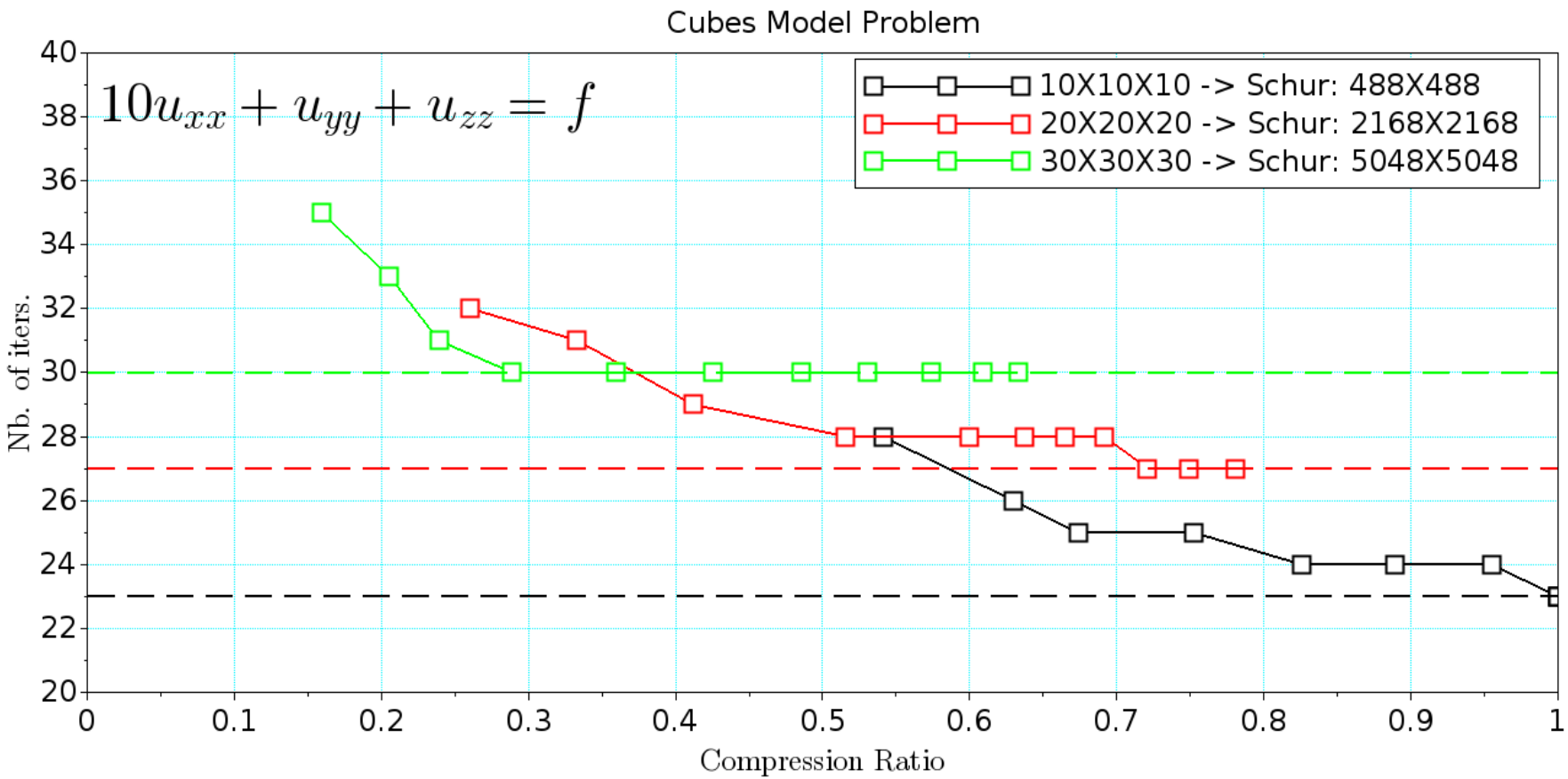
$$\text{Local Compression Ratio} = \frac{(m+n)k}{mn}$$



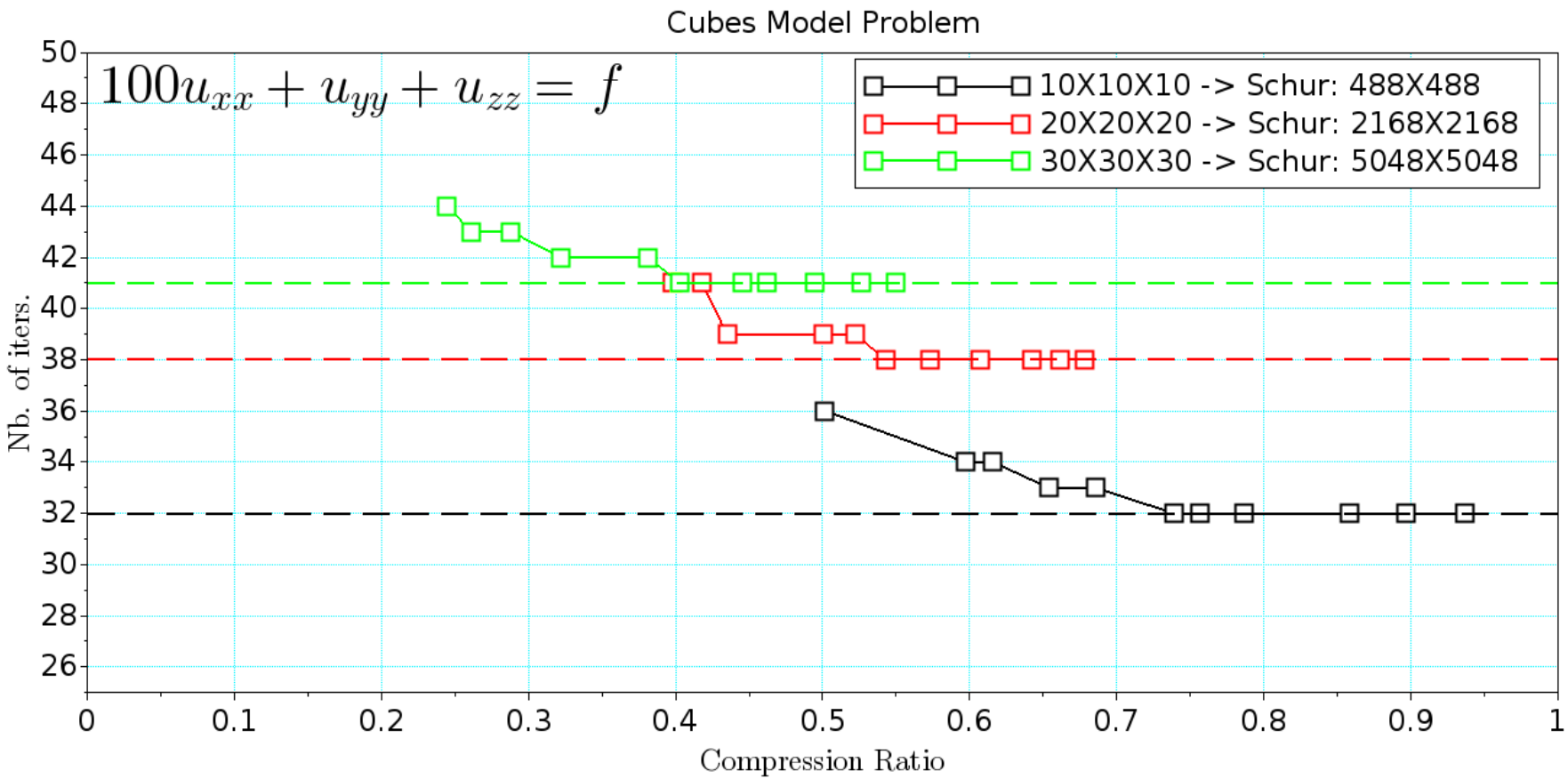
Results/Cubes Model



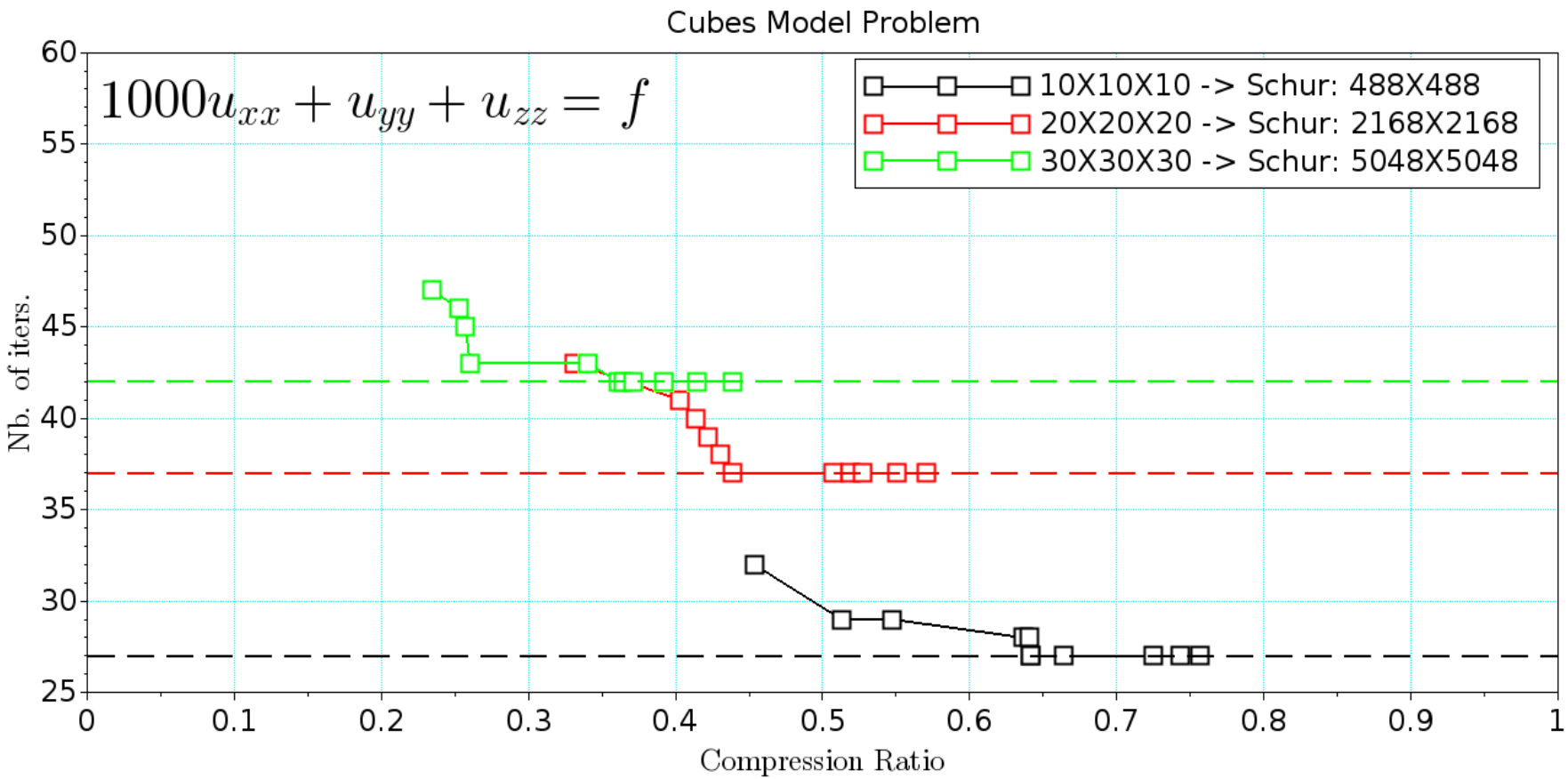
Results/Cubes Model



Results/Cubes Model



Results/Cubes Model



Conclusions

Next steps

- More challenging problems: Helmholtz, non canonical
- Comparison with ILU

Conclusions

Next steps

- More challenging problems: Helmholtz, non canonical
- Comparison with ILU

Long term

- replace PaStiX and MUMPS by fast H2 solver capable of generating compressed Schur complements.

Conclusions

Next steps

- More challenging problems: Helmholtz, non canonical
- Comparison with ILU

Long term

- replace PaStiX and MUMPS by fast H2 solver capable of generating compressed Schur complements.

Thank You