



# On the Use of H-Matrix Arithmetic in PaStiX, A Preliminary Study

June 26th, 2015 - Workshop Fast Direct Solvers

M. Faverge, G. Pichon, P. Ramet, J. Roman [Inria Bordeaux and Labri]  
A. Aminfar, E. Darve, C. Dudley [Stanford University]

## Outline

- ① Sparse Direct Solvers
- ② Introducing H-Matrix in PaStiX
- ③ Complexity Study
- ④ Black Box Approach
- ⑤ Current Experiments Towards H-PaStiX

# 1

## Sparse Direct Solvers

## Context

### Problem - Solve $Ax = b$

- Cholesky: factorize  $A = LL^T$  (symmetric pattern ( $A + A^T$ ) for  $LU$ )
- Solve  $Ly = b$
- Solve  $L^T x = y$

### Sparse Direct Solvers: PaStiX approach

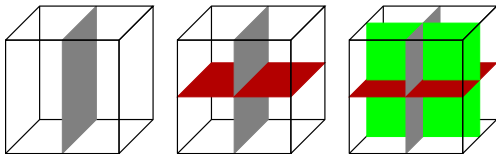
1. Order unknowns to minimize the fill-in
2. Compute a symbolic factorization to build  $L$  structure
3. Factorize the matrix in place on  $L$  structure
4. Solve the system with forward and backward triangular solves

## Nested Dissection

Considering a graph  $G = (V, E, \sigma_p)$   
 $V$ : vertexes,  $E$ : edges,  $\sigma_p$ : unknowns permutation

Algorithm to compute  $\sigma_p$

1. Partition  $V = A \cup B \cup C$
2. Order  $C$  with larger numbers:  $V_A < V_B < V_C$
3. Apply the process recursively on  $A$  and  $B$

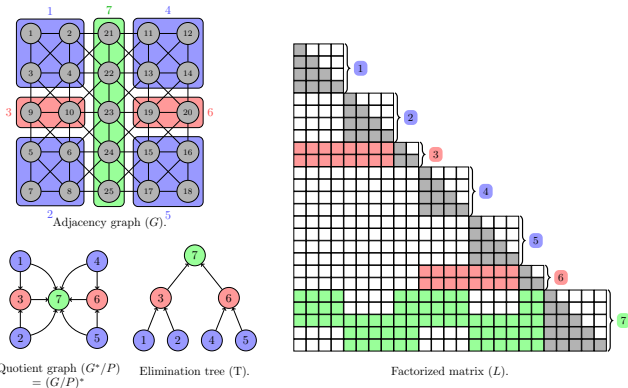


Three-levels of nested dissection on a regular cube

# Symbolic Factorization

## General approach

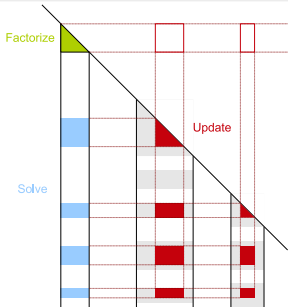
1. Build a partition with the nested dissection process
2. Compress information on data blocks
3. Compute elimination tree thanks to the quotient graph



# Numerical Factorization

Algorithm to eliminate the block column  $k$

1. Factorize the diagonal block
2. Solve off-diagonal blocks in the current column (TRSM)
3. Update the underlying matrix with the column's contribution (GEMM)



## Update

- Compacted matrix-matrix product
- Update divided into the number of off-diagonal blocks receiving contributions

# Motivations

## Cost of Direct Solvers

- Expensive with respect to iterative solvers
- Burden on scalability, especially because of memory consumption
- More robust, and allow to tackle hard problems

## Low-rank strategies

- Compress some information in the direct solver
- Provide a “good” preconditionner for iterative methods

Objective: target large difficult problems with a low-rank strategy that keeps the general idea of a supernodal approach



# 2

## Introducing H-Matrix in PaStiX

# Scenario

## Steps

1. Use direct approach on small supernodes
2. Compress large supernodes: dense diagonal block and off-diagonal blocks
3. Compute a low-rank factorization with both dense and low-rank blocks
4. Use the resulting solution as a “good” preconditionner for iterative methods

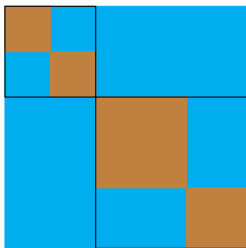
## Advantages

- Keep the inherent parallelism of the Right-Looking approach in PaStiX
- Do not update an H-structure with an H-structure

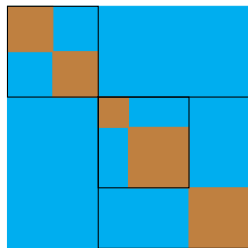
## HODLR Compression




Level 1




Level 2



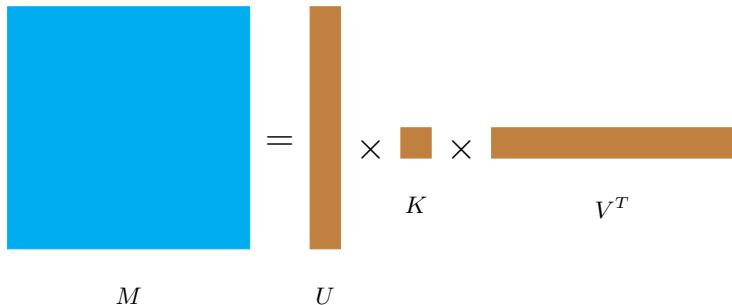
Level 3

 Full Rank

 Low Rank

Build an HODLR tree: dense diagonal blocks and low-rank off-diagonal blocks

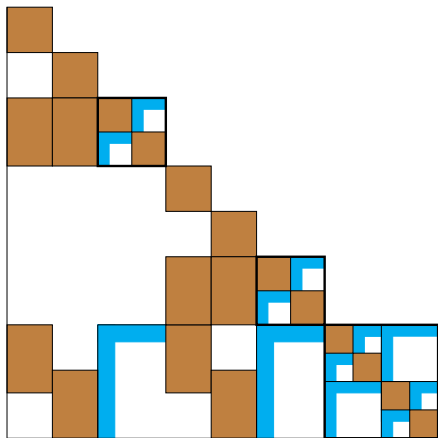
## Low-rank Compression



$$M \in \mathbb{R}^{n \times n}, U, V \in \mathbb{R}^{n \times r}, K \in \mathbb{R}^{r \times r}$$

Storage in  $2nr + r^2$  instead of  $n^2$   
Low rank compression with BDLR, ACA, SVD...

## Using HODLR in PaStiX



### Compress Information

- Large diagonal block: HODLR
- Large off-diagonal blocks: low-rank

### Underlying Contributions

- Dense to low-rank
- Dense to HODLR
- Low-rank to low-rank
- Low-rank to HODLR

## Open Issues

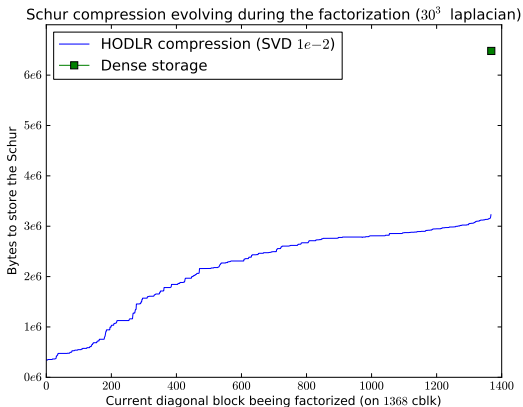
### HODLR management

- Factorization and Solve relying on HODLR kernels
- Update: has to be tuned for PaStiX
- Initialization: set the HODLR tree (median bisection?)

### Compression structures

- The rank may grow during the factorization. When to compress? in  $A$ ? after some steps?
- How to choose the low-rank accuracy?
- How to control the number of iterations?

# Compression evolution on the Last Supernode of a $30^3$ laplacian

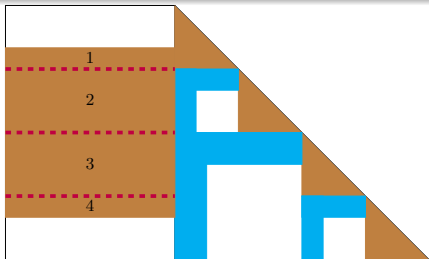


Compressed with HODLR using SVD and  $1e-2$  tolerance

## Apply an Update

### Applying an update to a HODLR structure

- Divide the contribution to fit the HODLR tree
- Update dense or low-rank structures instead of the complete HODLR structure
- Objective: build a low-rank symbolic structure



To enhance efficiency, it is important to study the structure of off-diagonal blocks contributing to a diagonal block



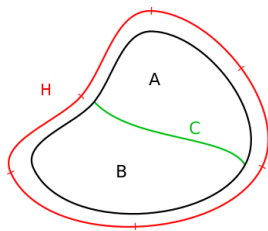
# 3

## Complexity Study

## Context of the Study (1)

### Conditions

- For bounded-density graphs [Miller, Valaris - 1991]
- $\bar{G} = (\bar{V}, \bar{E})$  with  $|\bar{V}| = p$
- Partition into  $\bar{V} = A \cup B \cup C$



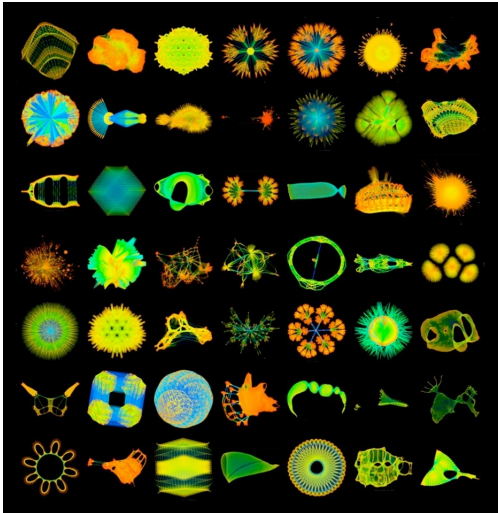
Partitioning a graph

### $p^\sigma$ -Separation Theorem [Lipton, Tarjan - 1979]

- $0 < \alpha < 1, \beta > 0, \frac{1}{2} \leq \sigma < 1$
- $|A| \leq \alpha p, |B| \leq \alpha p$
- $|C| \leq \beta p^\sigma$

## Context of the Study (2)

The University of Florida Sparse Matrix Collection



# Compression Ratio

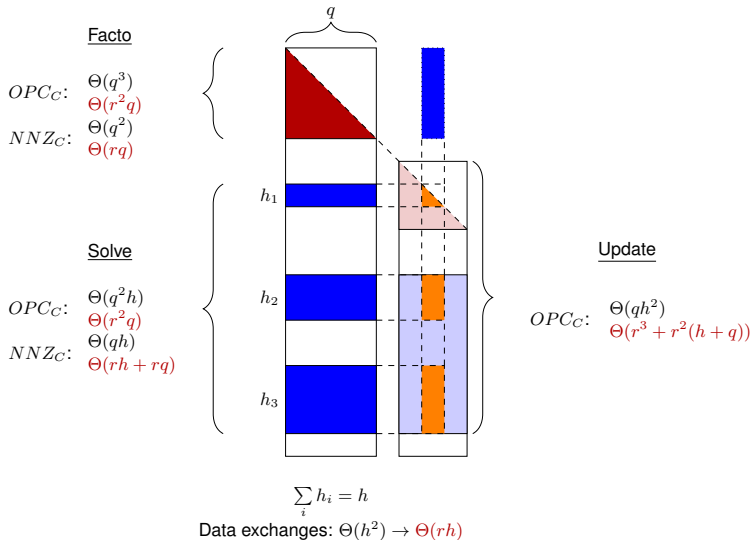
## Properties

- $|C| = q \leq \beta p^\sigma$
- $|H| = h = \Theta(q)$ : inner and outer parts are well balanced
- $r = \Theta(q^x) = \Theta(p^{\sigma x})$  with  $0 \leq x < 1$  adaptive

## HODLR

- AmirHossein Aminfar, Eric Darve: *A Fast and Memory Efficient Sparse Solver with Applications to Finite-Element Matrices*
- Storage:  $\Theta(rq)$  for a  $q$ -by- $q$  block of rank  $r$
- Factorization:  $\Theta(r^2q)$  for a  $q$ -by- $q$  block of rank  $r$
- Solve:  $\Theta(r^2h)$  when solving  $h$  unknowns with the previous factorization

# Complexity on a Block-Column - Dense / **Compressed**



## Update Low-Rank with Low-Rank (1)

### Evolution of a LR structure

- Start with sparse CSC format
- Compress with the corresponding lines/columns into LR (no cheap)
- Update this low-rank structure with low-rank contributions

## Update Low-Rank with Low-Rank (2)

$$A \leftarrow B + C$$

$$B = U_B V_B^T, C = U_C V_C^T$$

$$A = U_B V_B^T + U_C V_C^T = (U_B \quad U_C) \times \begin{pmatrix} V_B^T \\ V_C^T \end{pmatrix}$$

### QR factorizations

- $(U_B U_C) = Q_U R_U$
- $(V_B V_C) = Q_V R_V$
- Cost in  $\Theta(n_z r^2)$  with  $r$  the rank and  $n_z$  the number of non-zeros

## Update Low-Rank with Low-Rank (3)

$$A = (Q_U R_U) \times (Q_V R_V)^T$$

Build the new low-rank structure

1. Perform an SVD of  $R_U R_V^T = u \sigma v^T$
2. Keep  $r$  largest singular values
3.  $A = (Q_U u \sigma^{\frac{1}{2}})(Q_V v \sigma^{\frac{1}{2}})^T$

Result

- Computational cost in  $\Theta(r^3 + n_z r^2)$
- Memory requirement in  $\Theta(n_z r)$



## Complexity on a Block-Column

### In practice

- For a  $d$ -dim bounded-density graph,  $\sigma = \frac{d-1}{d}$
- The rank  $r = \Theta(q^x)$  is taken with  $x = \frac{d-2}{d-1}$

$$\text{compl}(\bar{G}, H) \leq \text{contrib}_C + \text{compl}(A, H_A) + \text{compl}(B, H_B)$$

Overall complexity depending on  $\text{contrib}_C = \Theta(p^y)$  [Charrier, Roman - 1989]

- $y < 1 \rightarrow \Theta(n)$
- $y = 1 \rightarrow \Theta(n \ln(n))$
- $y > 1 \rightarrow \Theta(n^y)$

## Results for the Factorization on general 2D/3D graphs

We consider a subgraph of size  $p$  with a separator  $C$ , while the original matrix is of size  $n$ .

### PaStiX 2D

- $OPC$ :  $\Theta(n^{\frac{3}{2}})$
- $NNZ$ :  $\Theta(n \ln(n))$

### H-PaStiX 2D - $\sigma = \frac{1}{2}$

- $OPC_C$ :  $\Theta(p^{\frac{1}{2}}) \rightarrow \Theta(n)$
- $NNZ_C$ :  $\Theta(p^{\frac{1}{2}}) \rightarrow \Theta(n)$

### PaStiX 3D

- $OPC$ :  $\Theta(n^2)$
- $NNZ$ :  $\Theta(n^{\frac{4}{3}})$

### H-PaStiX 3D - $\sigma = \frac{2}{3}$

- $OPC_C$ :  $\Theta(p^{\frac{4}{3}} + p) \rightarrow \Theta(n^{\frac{4}{3}} + n \ln(n))$
- $NNZ_C$ :  $\Theta(p) \rightarrow \Theta(n \ln(n))$

Similar results for HSS (Randomized Sparse Direct Solvers, by Jianlin Xia), except for OPC in 3D: HSS is in  $\Theta(n^{\frac{4}{3}} \ln(n))$

## Results for the Overall Solver on general 3D Graphs

If the resulting solver is used as a preconditioner for iterative methods

### Operations

- Factorization in  $\Theta(n^{\frac{4}{3}})$
- Solve in  $\Theta(n \ln(n))$

### Overall cost

- Considering  $m$  iterations in the iterative refinement
- Cost:  $k_1 \times n^{\frac{4}{3}} + k_2 \times n \ln(n)$
- The number of iterations should be lower than  $\Theta(\frac{n^{\frac{1}{3}}}{n \ln(n)})$  if one wants to keep a complexity in  $\Theta(n^{\frac{4}{3}})$

# 4

## Black Box Approach

## Idea

### Compression library dependency

- Require a dense library with a compression strategy (HODLR,  $\mathcal{H}$ ,  $\mathcal{H}^2$ , HSS...)
- Need compression/factorization/solve operations
- Do not require to compute an  $LU$  factorization in the compressed format

Implement H-PaStiX in order to depend on an hierarchical compression method, but without modifying the original implementation

## Formulas

### Classic LU

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \times \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = \begin{pmatrix} L_{11} \times U_{11} & L_{11} \times U_{12} \\ L_{21} \times U_{11} & L_{21} \times U_{12} + L_{22} \times U_{22} \end{pmatrix}$$

### H-LU

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11}^* & 0 \\ L_{21} & A_{22}^* \end{pmatrix} \times \begin{pmatrix} I & U_{12} \\ 0 & I \end{pmatrix} = \begin{pmatrix} A_{11}^* & A_{11}^* \times U_{12} \\ L_{21} & L_{21} \times U_{12} + A_{22}^* \end{pmatrix}$$

1. Factorize  $A_{11}$  in  $A_{11}^*$
2.  $L_{21} = A_{21}$
3. Solve  $A_{11}^* * U_{12} = A_{12}$
4. Compute  $A_{22}^{k+1} = A_{22}^k - L_{21} * U_{21}$

Solve on  $A_{kk}$  instead of solves on  $L_{kk}$  and  $L_{kk}$ .

Do not require to perform a LU decomposition in compressed format

# 5

## Current Experiments Towards H-PaStiX

# Experiments with the Schur Complement

## Context

- Compress only the last diagonal block
- Proof of concept
- Defining the interface to an H-Matrix library

## Strategy

1. Eliminate the first block-columns
2. Apply updates to the last supernode (in dense)
3. Compress this supernode
4. Factorize this supernode
5. In the forward/backward steps, replace the two solves on the last supernode by a single HODLR solve
6. Refine with GMRES



## Experiments with a regular grid (3D laplacian)

Method	Tolerance	Compression	Guess	Nb iterations	Norm
ACA	1e-1	4.25	1.76e-01	20	1.01e-08
	1e-2	2.37	1.55e-01	20	1.93e-12
	1e-3	1.61	1.16e-01	12	1.84e-13
	1e-4	1.36	5.50e-02	8	6.82e-13
	1e-5	1.26	3.37e-05	3	1.30e-14
	1e-6	1.18	1.17e-06	3	3.87e-15
BDLR	1e-1	2.48	3.12e-02	10	2.16e-13
	1e-2	2.00	9.31e-03	7	1.87e-14
	1e-3	1.65	1.31e-03	5	2.03e-14
	1e-4	1.42	1.40e-04	4	3.41e-15
	1e-5	1.30	7.84e-06	3	4.16e-15
	1e-6	1.23	1.13e-06	3	2.89e-15
SVD	1e-1	2.99	2.45e-02	9	1.85e-13
	1e-2	2.13	3.81e-03	6	3.65e-14
	1e-3	1.72	2.26e-04	4	1.33e-14
	1e-4	1.48	1.77e-05	3	3.06e-14
	1e-5	1.34	1.19e-06	3	3.39e-15
	1e-6	1.26	9.49e-08	2	3.39e-14

## Experiments with a real-life matrix (audi)

Method	Tolerance	Compression	Guess	Nb iterations	Norm
ACA	1e-1	6.94	2.96e-03	20	4.94e-06
	1e-2	2.94	7.22e-04	20	3.68e-07
	1e-3	2.00	5.66e-05	12	8.50e-13
	1e-4	1.68	2.91e-06	5	4.36e-14
	1e-5	1.50	9.38e-08	3	1.94e-13
	1e-6	1.34	9.05e-09	2	5.87e-13
BDLR	1e-1	4.53	1.20e-03	20	1.84e-06
	1e-2	3.19	3.27e-04	20	1.41e-07
	1e-3	2.43	9.77e-03	15	1.65e-13
	1e-4	1.97	4.02e-06	7	1.68e-13
	1e-5	1.70	3.13e-07	4	3.50e-13
	1e-6	1.52	1.53e-08	3	4.40e-14
SVD	1e-1	5.07	7.48e-04	20	2.16e-07
	1e-2	3.24	1.34e-04	20	1.65e-10
	1e-3	2.45	9.74e-06	7	9.55e-13
	1e-4	1.99	8.27e-07	4	2.75e-13
	1e-5	1.73	8.42e-08	3	4.61e-14
	1e-6	1.54	8.55e-09	2	3.14e-13

## Numerical Results

Accuracy of the resulting solution depends on:

- The compression method (SVD is the best)
- The HODLR tree (structure and depth)
- The given tolerance to HODLR code
- The number of iterations allowed in the iterative process

Preliminary experiments

- SVD better than BDLR, better than ACA
- Small number of iterations to reach PaStiX original accuracy

# Conclusion

## Perspectives

- Linear solver for  $2D$  general graphs
- A  $\Theta(n^{\frac{4}{3}})$  solver for  $3D$  general graphs
- Memory improvements
- OPC gain, but probably loss in effectiveness (BLAS)
- Large challenging problems

## Future Work

- Develop black-box approach
- Study HODLR tree with respect to off-diagonal contributions
- Verify the condition  $x = \frac{d-2}{d-1}$  in practice

**Thanks !**

## Complexity on a Block-Column

### Cost

- $q = \Theta(p^\sigma) = h, r = \Theta(p^{\sigma x})$
- $OPCC: \Theta(r^2(q + h) + r^3) = \Theta(p^{\sigma(2x+1)} + p^{3\sigma x})$
- $NNZ_C: \Theta(r(q + h)) = \Theta(p^{\sigma(x+1)})$

### In practice

- For a  $d$ -dim bounded-density graph,  $\sigma = \frac{d-1}{d}$
- The rank  $r = \Theta(q^x)$  is taken with  $x = \frac{d-2}{d-1}$
- Exponents for  $OPCC: \sigma(2x + 1) = \frac{3d-5}{d}, 3\sigma x = \frac{3d-6}{d}$
- Exponents for  $NNZ_C: \sigma(x + 1) = \frac{2d-3}{d}$

## Complexity: HODLR vs Randomized HSS

Dimension	Metric	HODLR	HSS	Classic
2D	<i>NNZ</i>	$\Theta(n)$	$\Theta(n)$	$\Theta(n \ln(n))$
	<i>OPC</i>	$\Theta(n)$	$\Theta(n)$	$\Theta(n^{\frac{3}{2}})$
3D	<i>NNZ</i>	$\Theta(n \ln(n))$	$\Theta(n \ln(n))$	$\Theta(n^{\frac{4}{3}})$
	<i>OPC</i>	$\Theta(n^{\frac{4}{3}})$	$\Theta(n^{\frac{4}{3}} \ln(n))$	$\Theta(n^2)$