Sparse Gaussian Elimination Modulo p: an Update

Charles Bouillaguet [1] Claire Delaplace [2]

[1,2]CRIStAL, Université de Lille

[2]Université de Rennes 1, IRISA

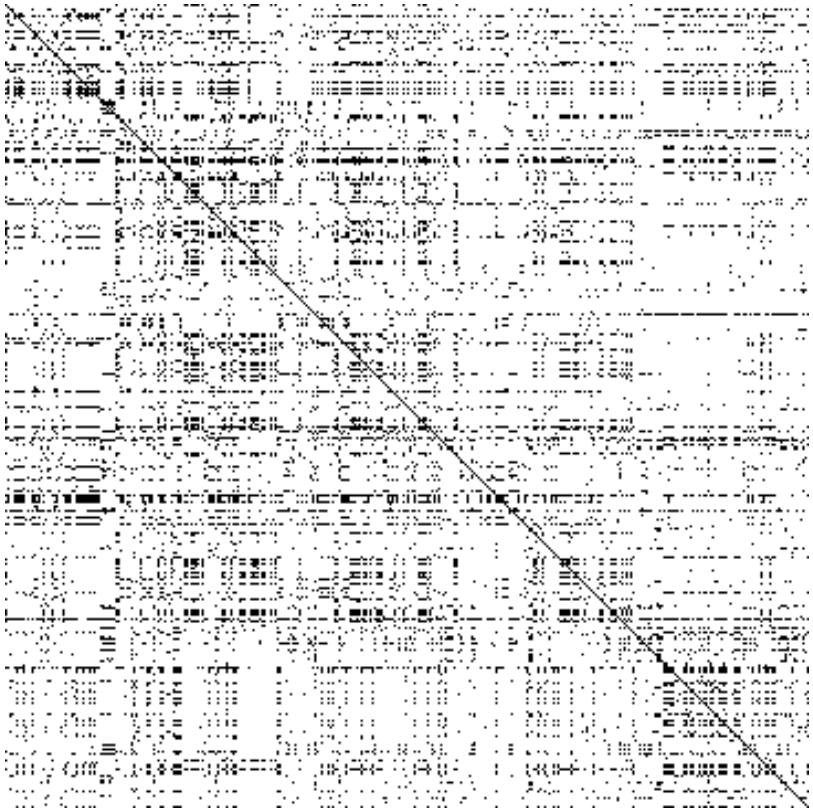
CASC Conference, September 2016

Background

Sparse Linear Algebra
Modulo p (coefficients : int)

Operations

- Rank
- Linear systems
- Kernel
- etc...

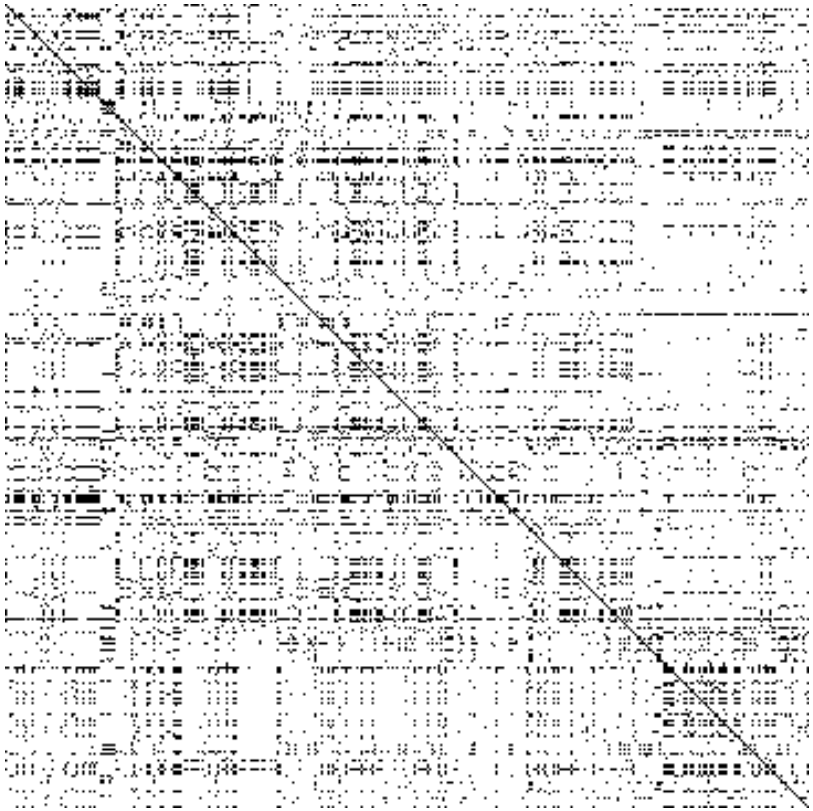


Background

Sparse Linear Algebra
Modulo p (coefficients : int)

Operations

- Rank
- Linear systems
- Kernel
- etc...



Two families of Algorithms

- **Direct methods** (Gaussian Elimination, LU, ...): Numerical World
- **Iterative methods** (Wiedemann, ...): Linear Algebra

Related Work

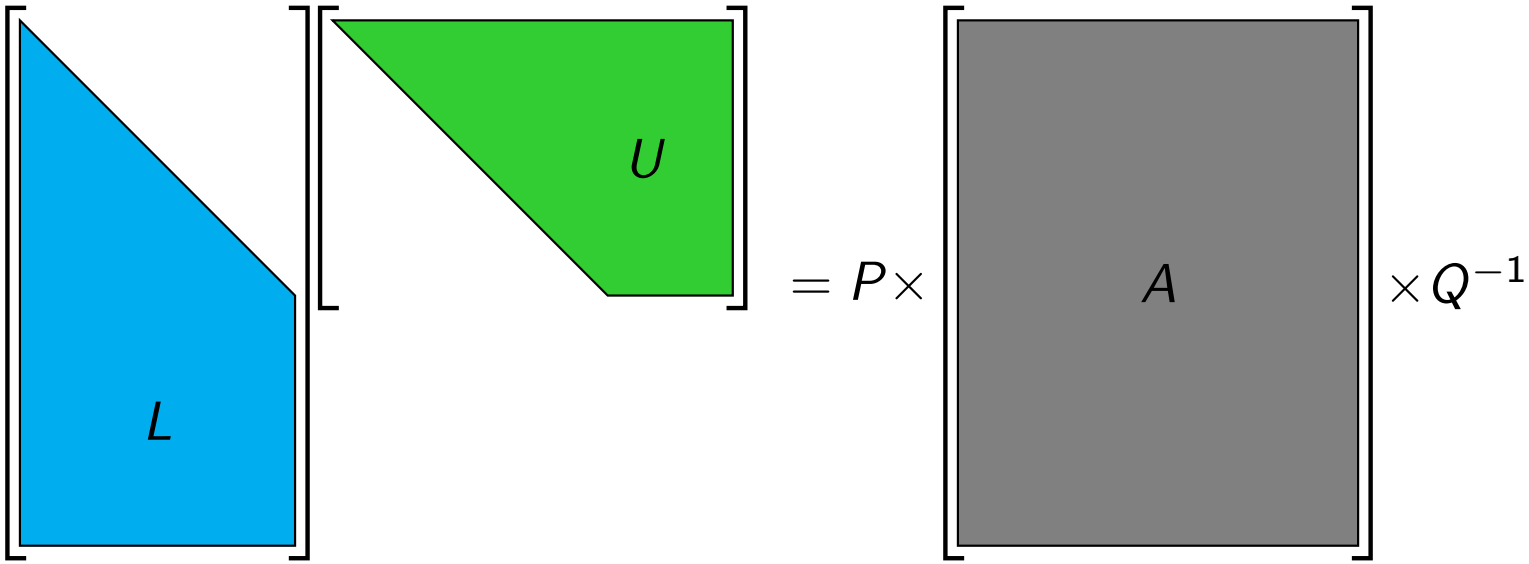
Algorithms

- Comparison between a sparse gaussian elimination and the Wiedmann algorithm: [Dumas & Villard 02]
- Direct methods in the numerical world (e.g. [Davis 06])
- Pivots selection heuristic for Gröbner Basis Matrices [Faugère & Lacharte 10]

Software

- Exact: *not much* (LinBox, GBLA (*Gröbner basis*), MAGMA)
- Numeric: *many* (SuperLU, UMFPACK, ...)

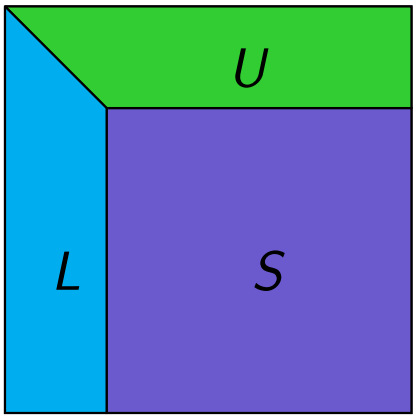
PLUQ Factorization



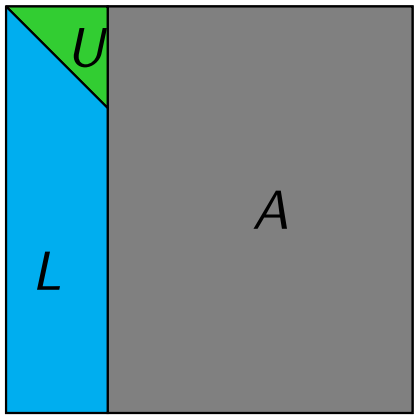
- L has unit diagonal
- U has non zero diagonal
- A can be rectangular
- A can be rank deficient

Right-looking and Left-looking LU

Usual right-looking Algorithm:

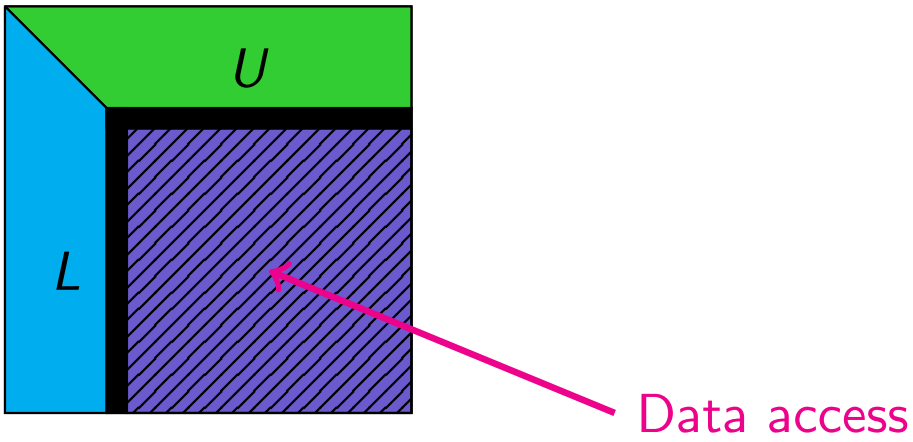


Left-looking GPLU Algorithm
[Gilbert & Peierls 88]

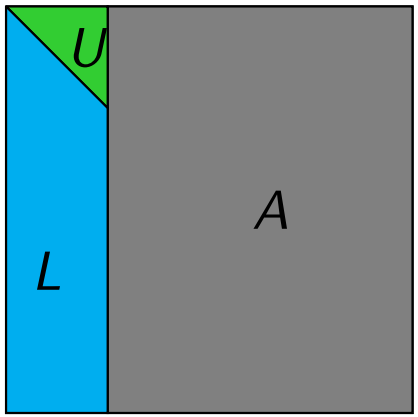


Right-looking and Left-looking LU

Usual right-looking Algorithm:

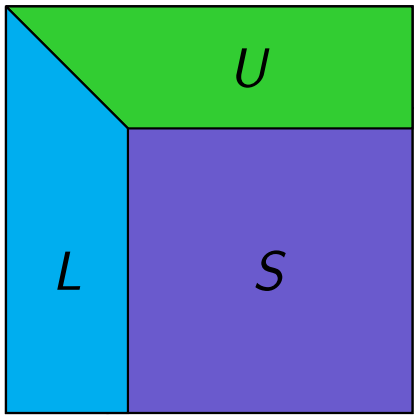


Left-looking GPLU Algorithm
[Gilbert & Peierls 88]

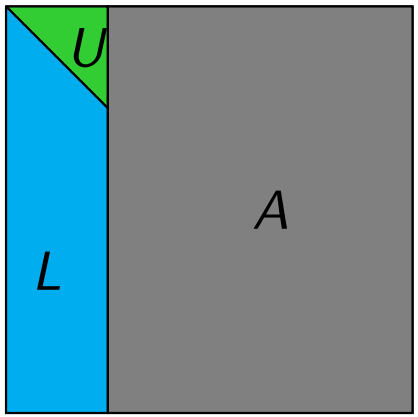


Right-looking and Left-looking LU

Usual right-looking Algorithm:

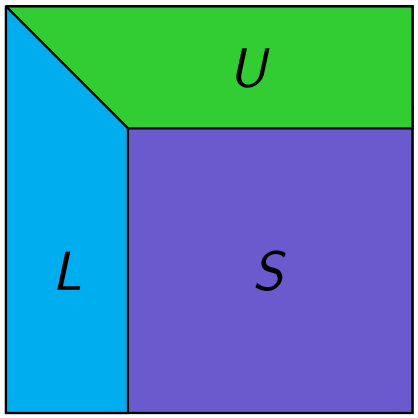


Left-looking GPLU Algorithm
[Gilbert & Peierls 88]

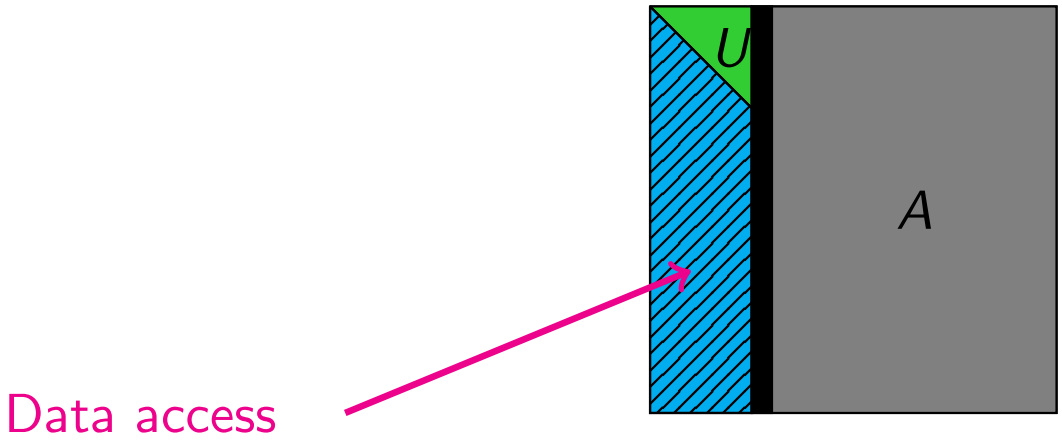


Right-looking and Left-looking LU

Usual right-looking Algorithm:

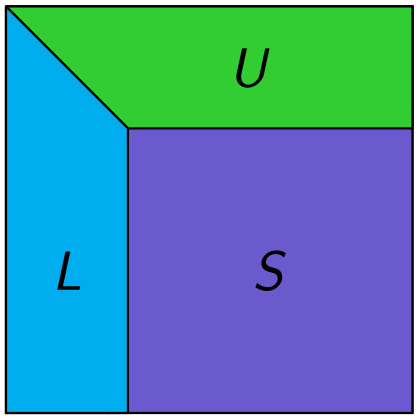


Left-looking GPLU Algorithm
[Gilbert & Peierls 88]

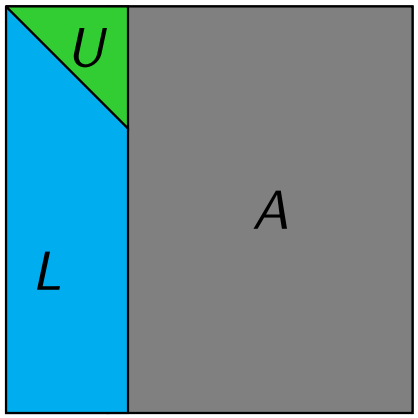


Right-looking and Left-looking LU

Usual right-looking Algorithm:



Left-looking GPLU Algorithm
[Gilbert & Peierls 88]



GPLU Algorithm: Elimination Step

Ignoring the permutations:

$$A = \begin{pmatrix} A_{00} & \mathbf{a}_{01} & A_{02} \\ \mathbf{a}_{10} & a_{11} & \mathbf{a}_{12} \\ A_{20} & \mathbf{a}_{21} & A_{11} \end{pmatrix} = \begin{pmatrix} L_{00} & & \\ \mathbf{l}_{10} & 1 & \\ L_{20} & \mathbf{l}_{21} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} U_{00} & \mathbf{u}_{01} & U_{02} \\ & u_{11} & \mathbf{u}_{12} \\ & & U_{22} \end{pmatrix}$$

where $(\mathbf{a}_{10}\ a_{11}\ \mathbf{a}_{12})$ is the k-th of A.

GPLU Algorithm: Elimination Step

Ignoring the permutations:

$$A = \begin{pmatrix} A_{00} & \mathbf{a}_{01} & A_{02} \\ \mathbf{a}_{10} & a_{11} & \mathbf{a}_{12} \\ A_{20} & \mathbf{a}_{21} & A_{11} \end{pmatrix} = \begin{pmatrix} L_{00} & & \\ \mathbf{l}_{10} & 1 & \\ L_{20} & \mathbf{l}_{21} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} U_{00} & \mathbf{u}_{01} & U_{02} \\ & u_{11} & \mathbf{u}_{12} \\ & & U_{22} \end{pmatrix}$$

where $(\mathbf{a}_{10}\ a_{11}\ \mathbf{a}_{12})$ is the k-th of A.

In particular:

$$(\mathbf{a}_{10}\ \mathbf{a}_{11}\ \mathbf{a}_{12}) = (\mathbf{l}_{10}\ u_{11}\ \mathbf{u}_{12}) \cdot \begin{pmatrix} U_{00} & \mathbf{u}_{01} & U_{02} \\ & 1 & \\ & & Id \end{pmatrix}$$

GPLU Algorithm: Application to Exact Linear Algebra

- Never been used before...
- We implemented it
- We benchmarked it against LinBox (sparse right-looking)

Our Benchmarks show:

- GPLU work best when U is very sparse
- Sometimes GPLU outperform the right-looking algorithm (often)
- Sometimes the right-looking algorithm outperform GPLU (less often)

$\Longrightarrow$ Can we take advantage of both methods?

Our Work: A New Hybrid Algorithm

Description

- Find pivots without performing any arithmetical operations.
- Compute the Schur complement S, using a left-looking algorithm.
- Compute the rank of S.

Our Work: A New Hybrid Algorithm

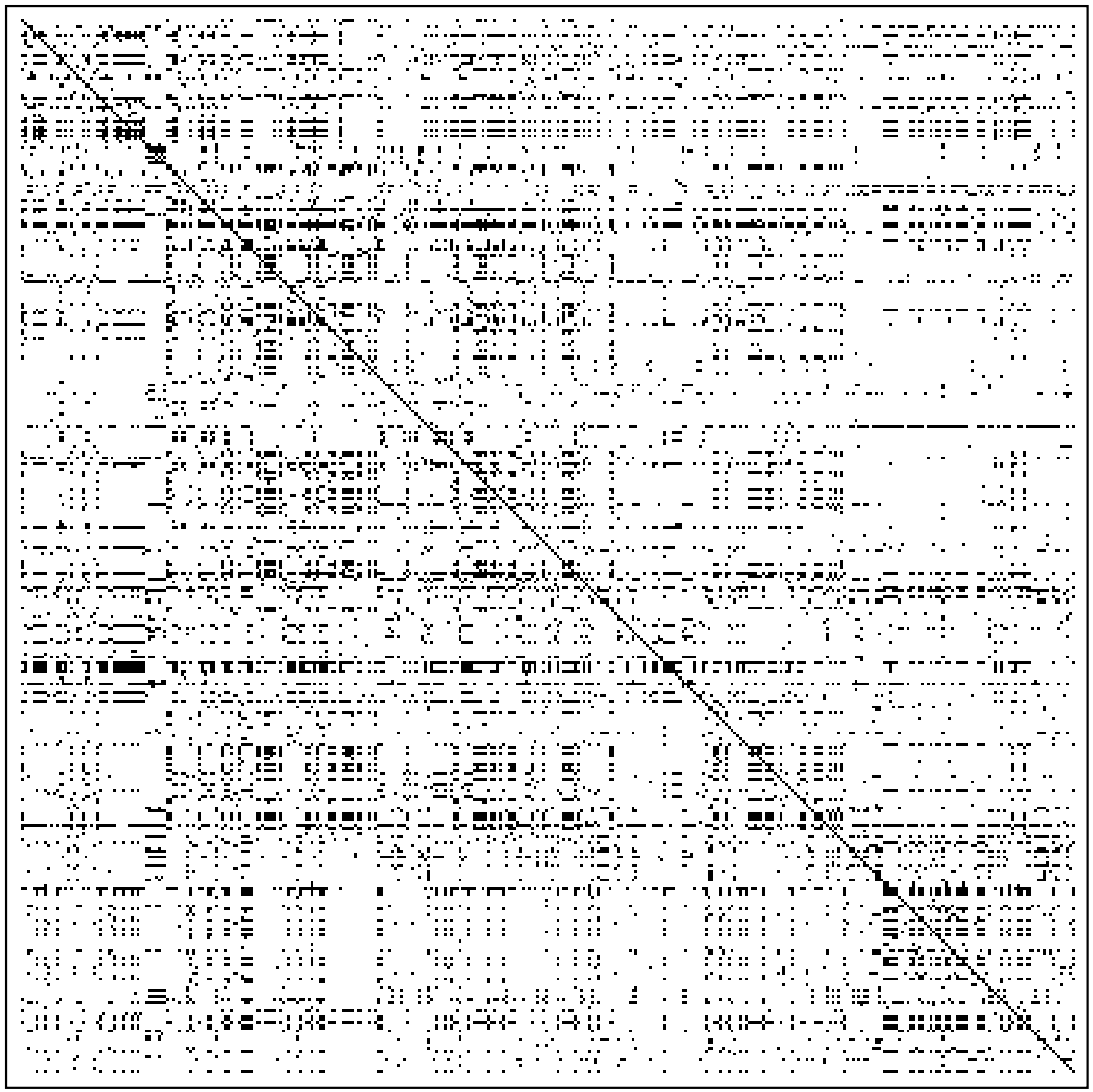
Description

- Find pivots without performing any arithmetical operations.
- Compute the Schur complement S, using a left-looking algorithm.
- Compute the rank of S.

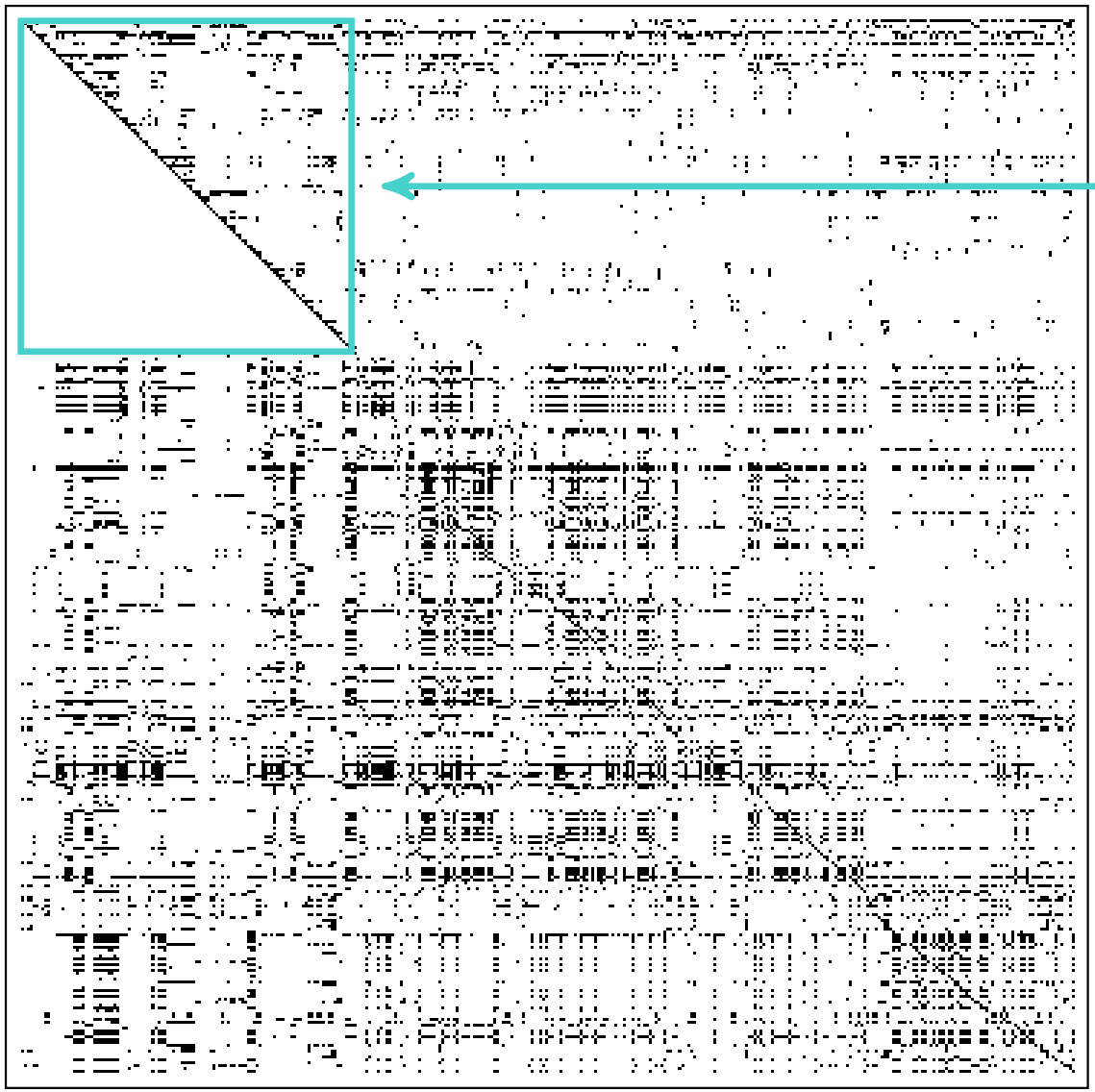
Rank of S

- Recurse
- Dense rank computation
- Wiedemann Algorithm
- ...

Example

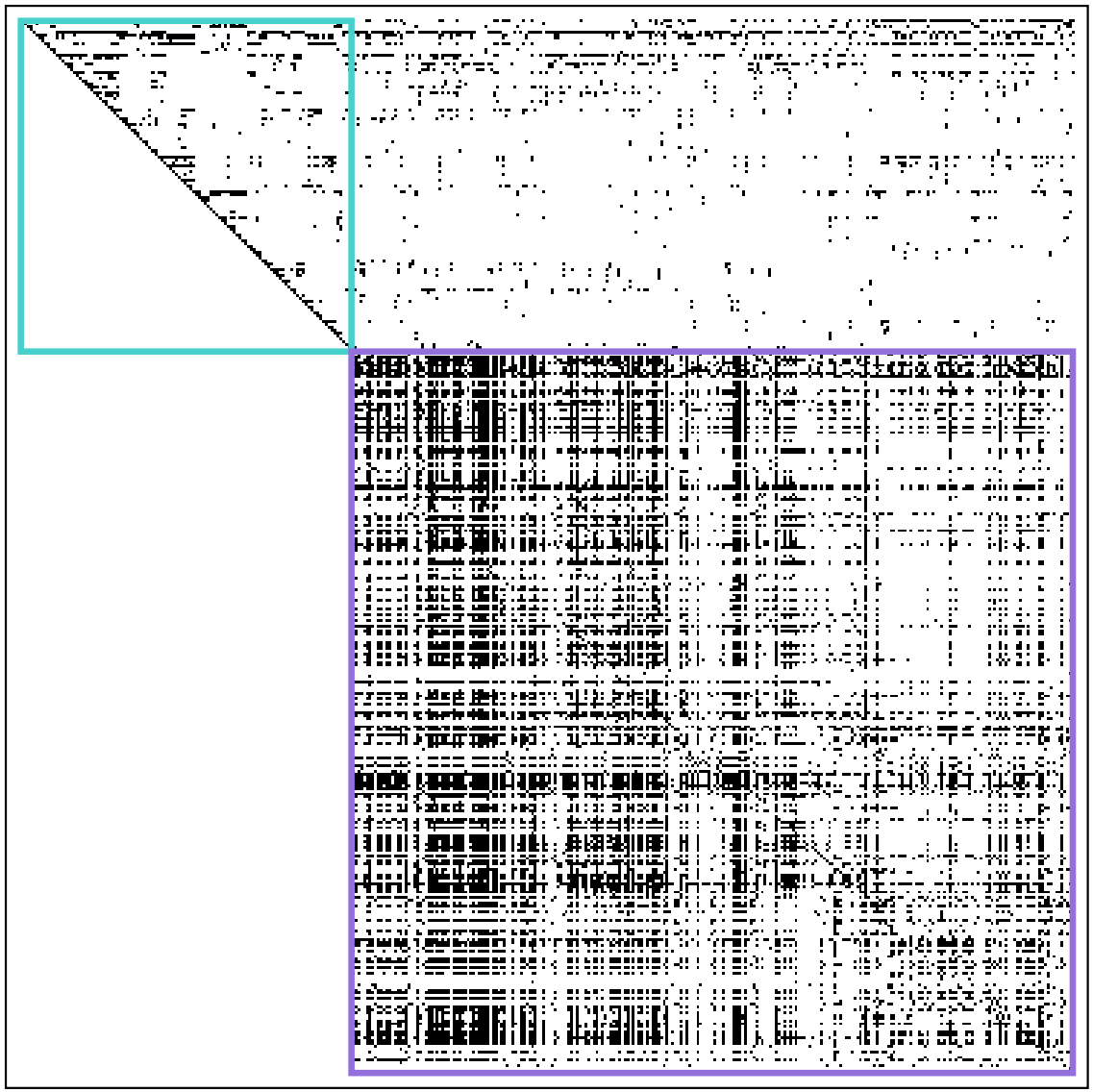


Example



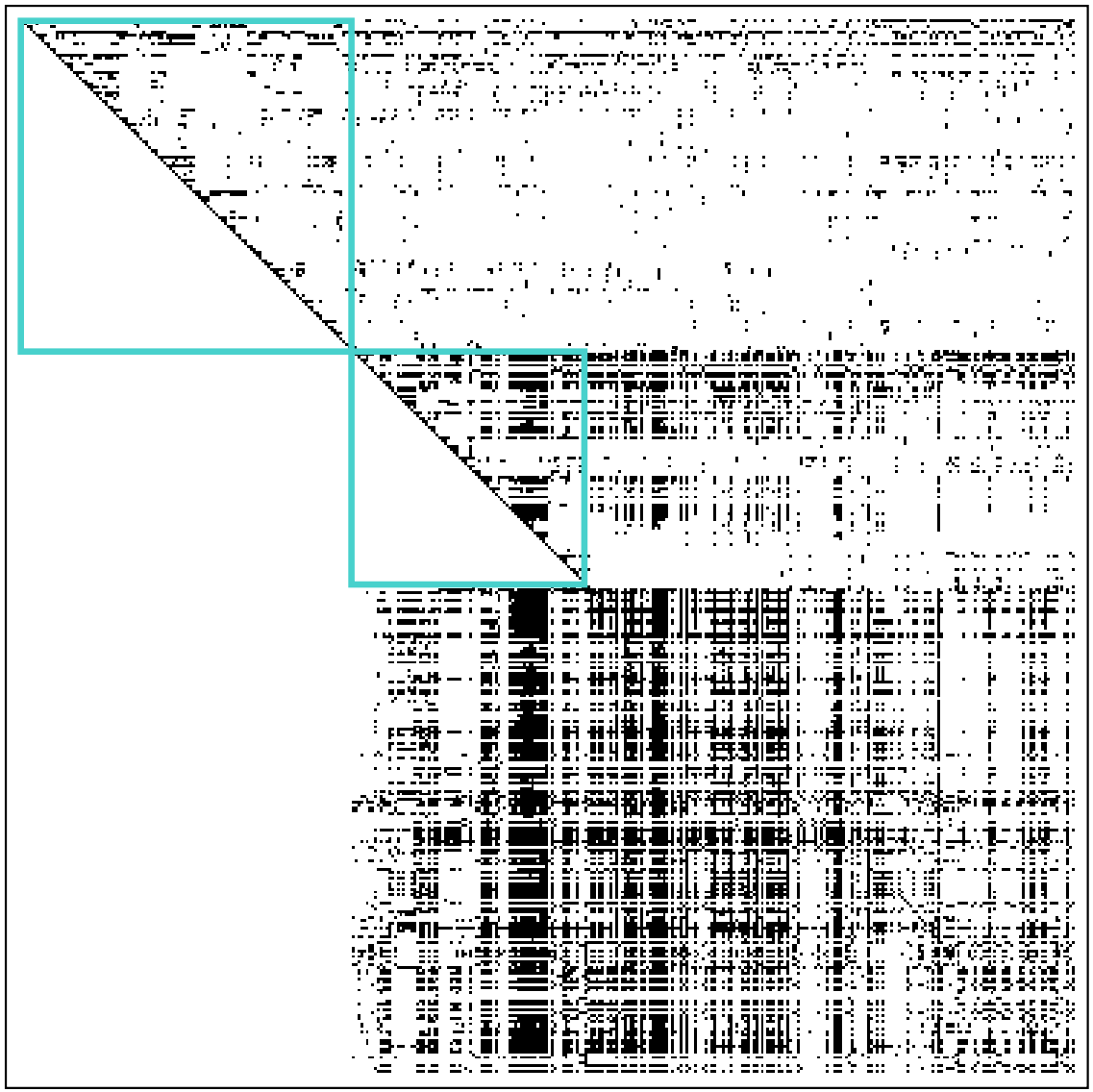
Set of pivots found without any arithmetical operations

Example

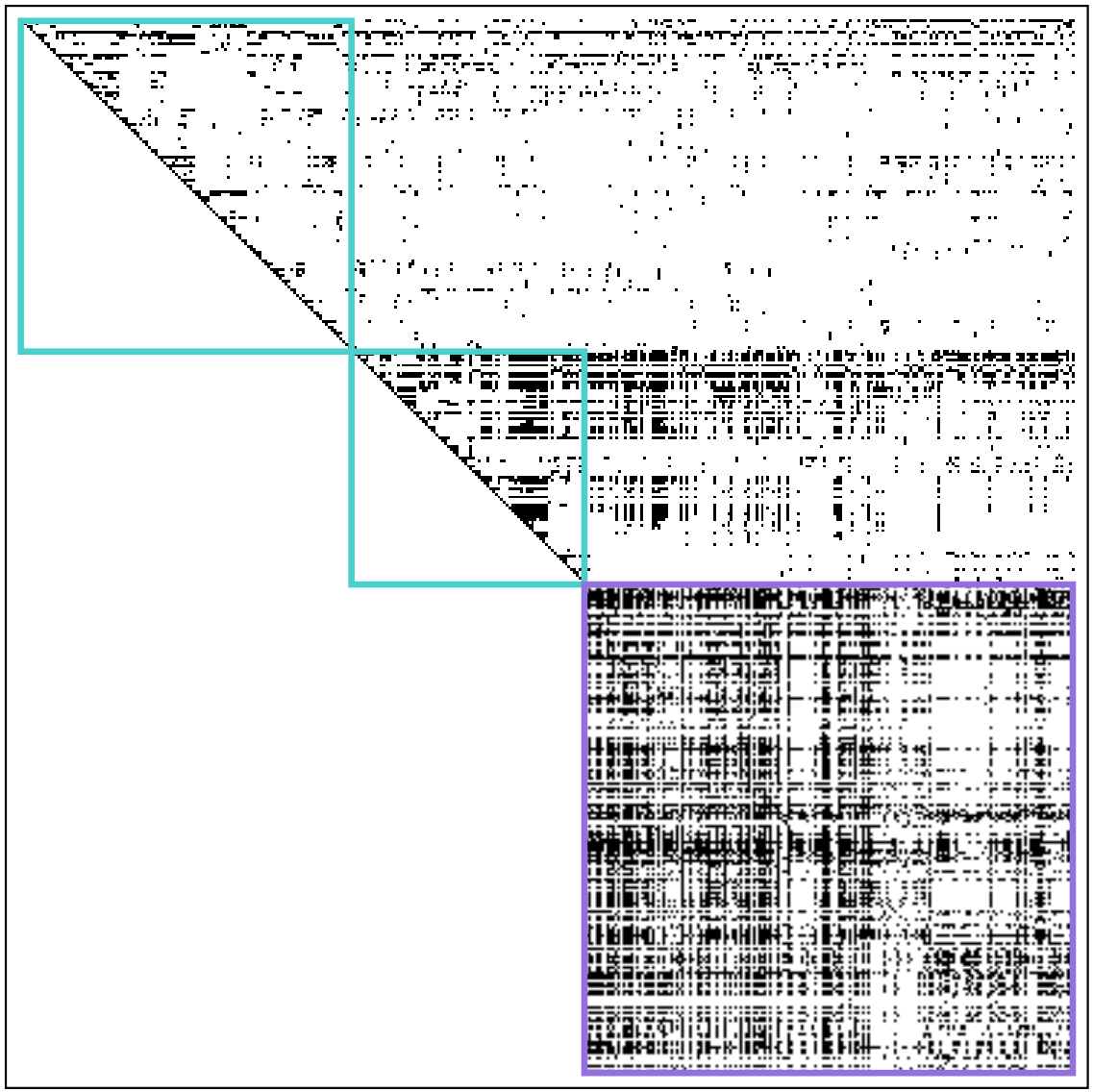


← Schur Complement

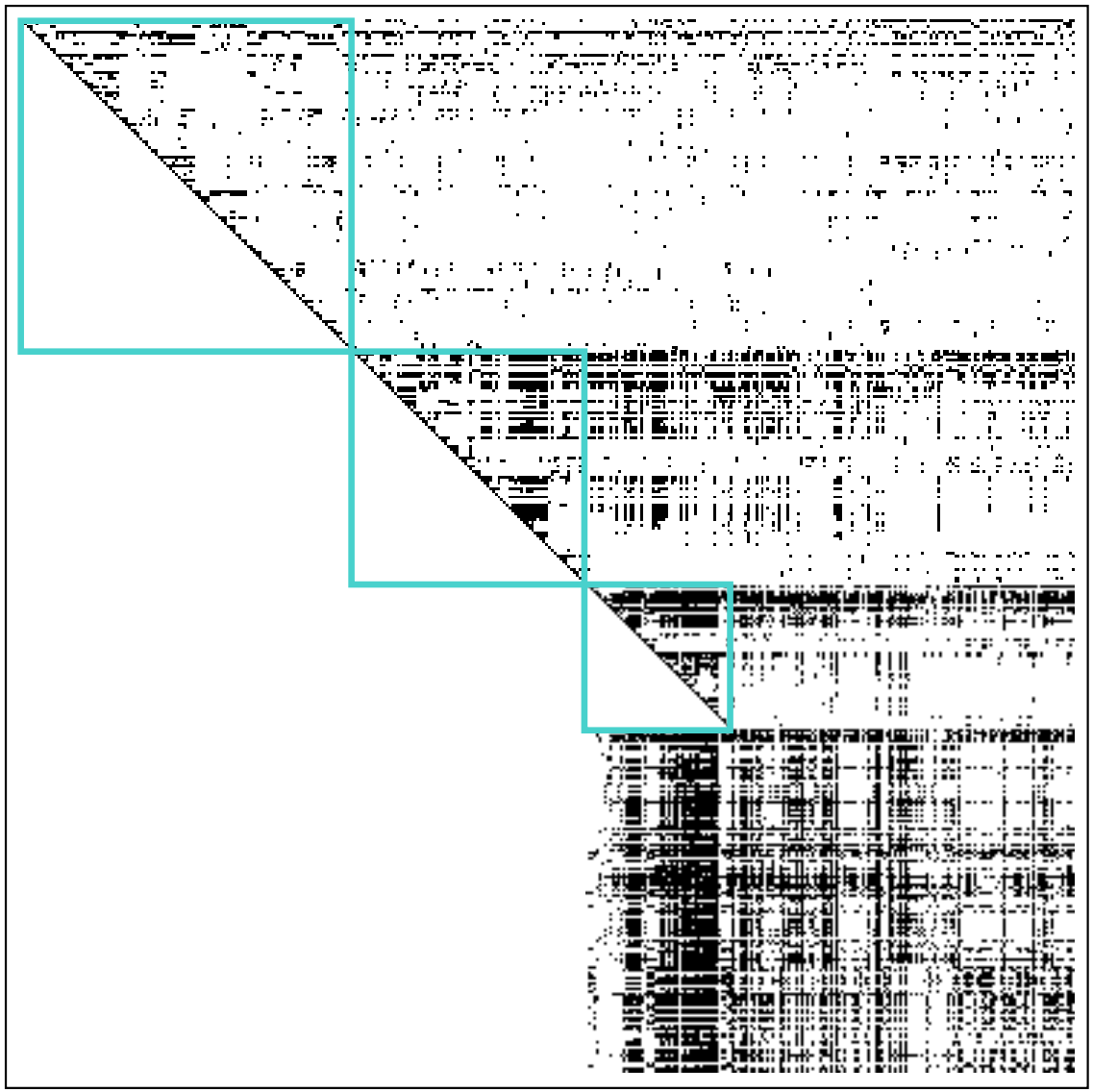
Example



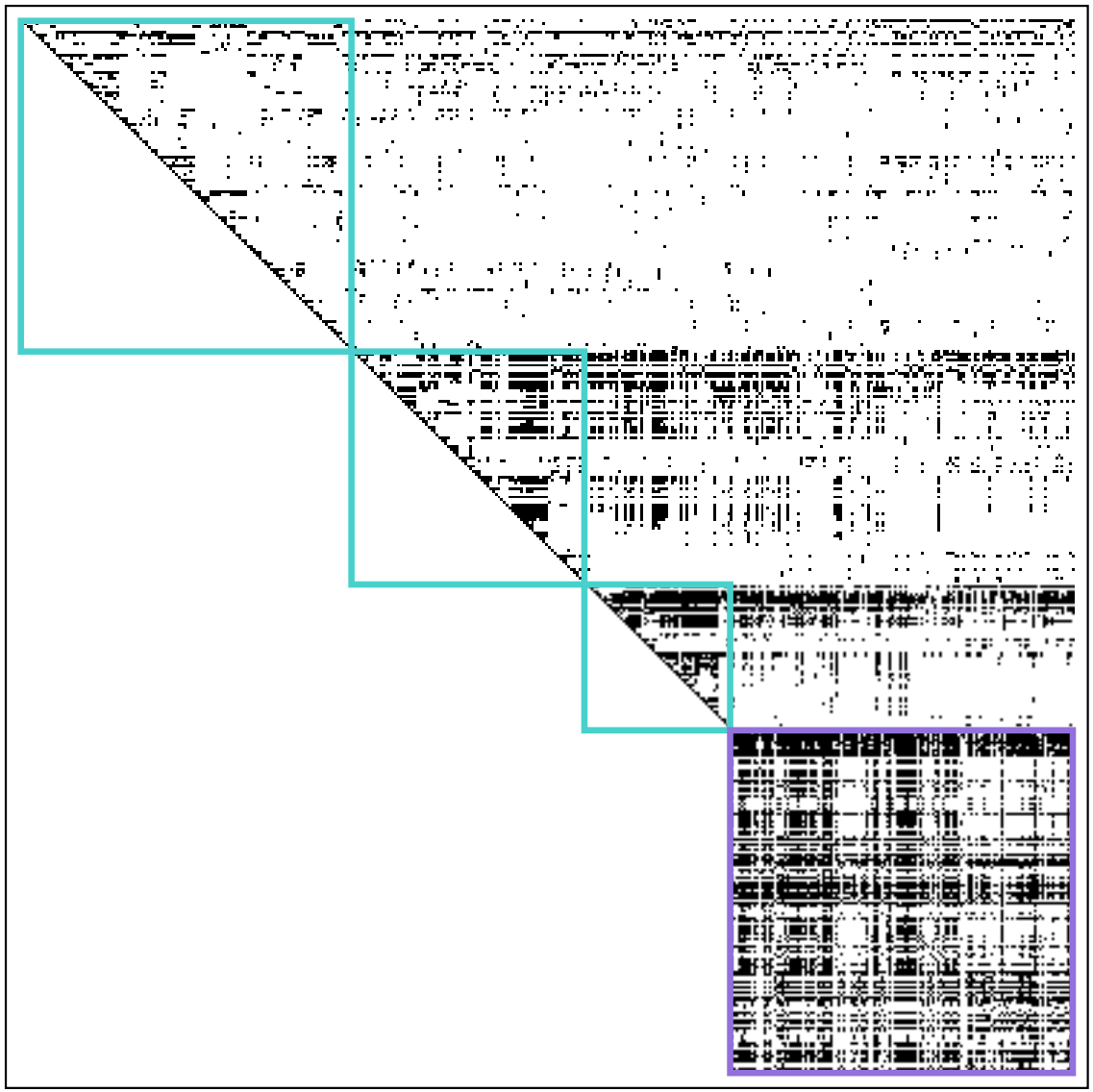
Example



Example

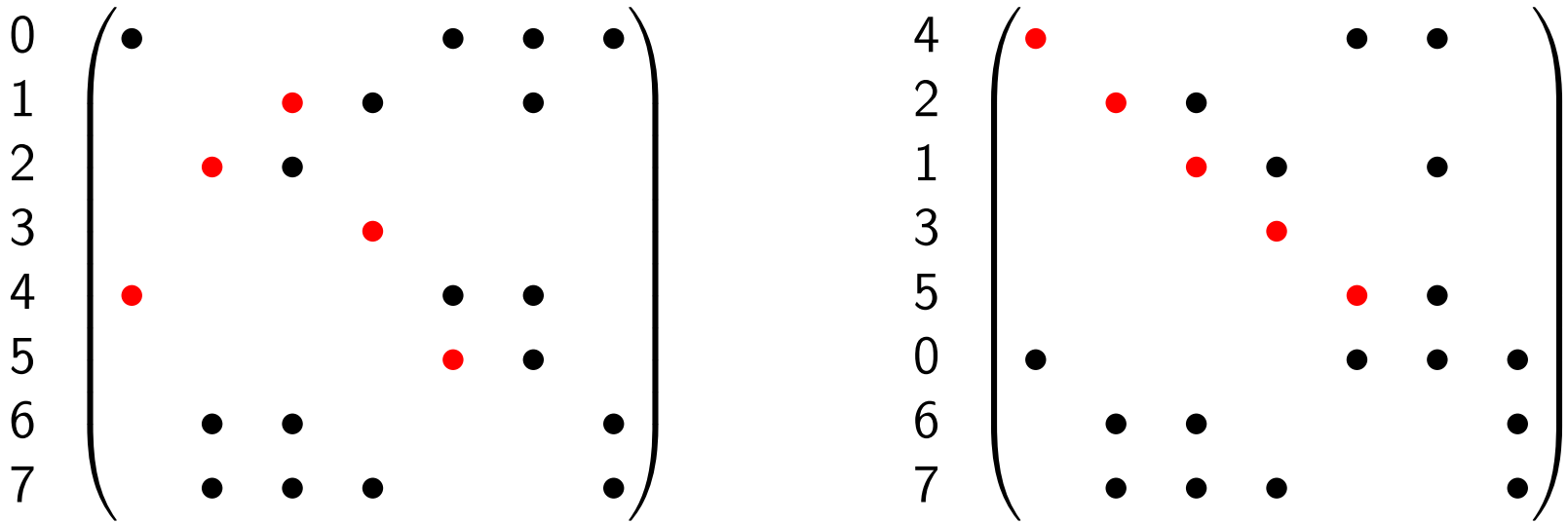


Example



← Parallelizable

Pivots Selection [Faugère & Lachartre 10]



Description

- Each row is mapped to the column of its leftmost coefficient.
- When several rows have the same leftmost coefficient, select the sparsest.
- Move the selected rows before the others and sort them by increasing position of the leftmost coefficient.

Schur Complement Computation

P denotes the permutation that pushes the pre-computed pivots in the top of A. Ignoring permutation over the columns of A:

$$PA = \begin{pmatrix} U_{00} & U_{01} \\ A_{10} & A_{11} \end{pmatrix} = \begin{pmatrix} Id & \\ L_{10} & L_{11} \end{pmatrix} \cdot \begin{pmatrix} U_{00} & U_{01} \\ & U_{11} \end{pmatrix}$$

Schur Complement Computation

P denotes the permutation that pushes the pre-computed pivots in the top of A. Ignoring permutation over the columns of A:

$$PA = \begin{pmatrix} U_{00} & U_{01} \\ A_{10} & A_{11} \end{pmatrix} = \begin{pmatrix} Id & \\ L_{10} & L_{11} \end{pmatrix} \cdot \begin{pmatrix} U_{00} & U_{01} \\ & U_{11} \end{pmatrix}$$

Definition

The **Schur Complement** S of PA with respect to U_{00} is given by :

$$S = A_{11} - A_{10} U_{00}^{-1} U_{01}$$

Schur Complement Computation

P denotes the permutation that pushes the pre-computed pivots in the top of A. Ignoring permutation over the columns of A:

$$PA = \begin{pmatrix} U_{00} & U_{01} \\ A_{10} & A_{11} \end{pmatrix} = \begin{pmatrix} Id & \\ L_{10} & L_{11} \end{pmatrix} \cdot \begin{pmatrix} U_{00} & U_{01} \\ & U_{11} \end{pmatrix}$$

Definition

The **Schur Complement** S of PA with respect to U_{00} is given by :

$$S = A_{11} - A_{10} U_{00}^{-1} U_{01}$$

Denote by $(\mathbf{a}_{i0}\ \mathbf{a}_{i1})$ the i-th row of $(A_{10}\ A_{11})$, and consider the following system :

$$(\mathbf{x}_0\ \mathbf{x}_1) \cdot \begin{pmatrix} U_{00} & U_{01} \\ & Id \end{pmatrix} = (\mathbf{a}_{i0}\ \mathbf{a}_{i1})$$

Schur Complement Computation

P denotes the permutation that pushes the pre-computed pivots in the top of A. Ignoring permutation over the columns of A:

$$PA = \begin{pmatrix} U_{00} & U_{01} \\ A_{10} & A_{11} \end{pmatrix} = \begin{pmatrix} Id & \\ L_{10} & L_{11} \end{pmatrix} \cdot \begin{pmatrix} U_{00} & U_{01} \\ & U_{11} \end{pmatrix}$$

Definition

The **Schur Complement** S of PA with respect to U_{00} is given by :

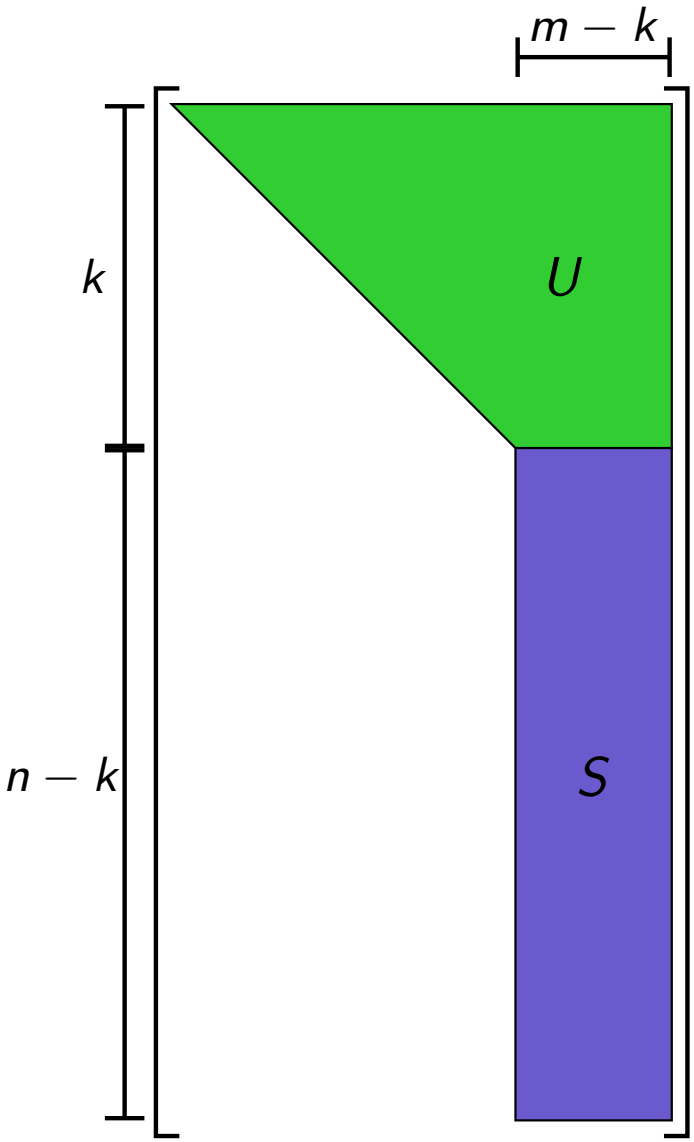
$$S = A_{11} - A_{10} U_{00}^{-1} U_{01}$$

Denote by $(\mathbf{a}_{i0}\ \mathbf{a}_{i1})$ the i-th row of $(A_{10}\ A_{11})$, and consider the following system :

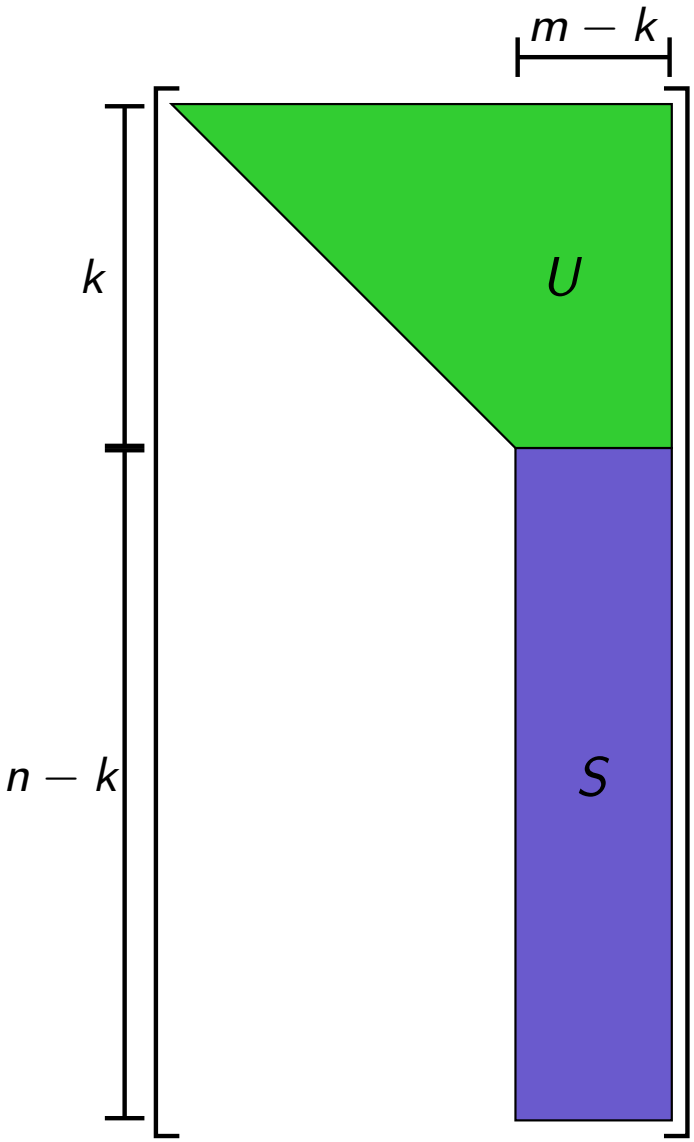
$$(\mathbf{x}_0\ \mathbf{x}_1) \cdot \begin{pmatrix} U_{00} & U_{01} \\ & Id \end{pmatrix} = (\mathbf{a}_{i0}\ \mathbf{a}_{i1})$$

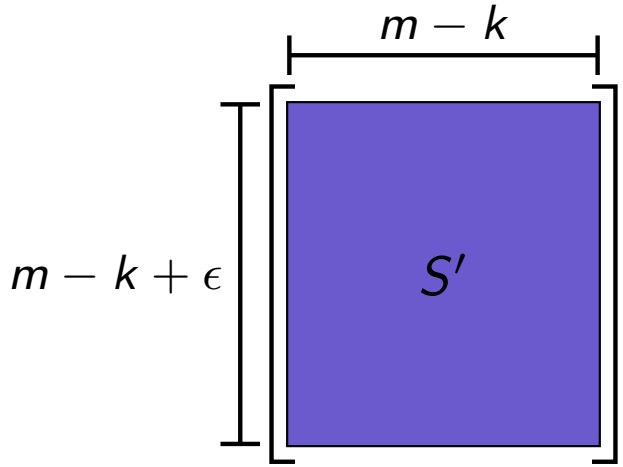
We obtain $\mathbf{x}_1 = \mathbf{a}_{i1} - \mathbf{a}_{i0} U_{00}^{-1} U_{01}$. $\mathbf{x}_1$ **is the i-th row of the** S

Tall and Narrow Schur Complement



Tall and Narrow Schur Complement





Idea

- Build a dense matrix S'
- The rows of S' are dense random linear combinations of the rows of S.
- Compute the rank of S'.

Results

Benchmark matrices: J.-G. Dumas Sparse Integer Matrix Collection (SIMC).
Experiment carried on an Intel Core i7-3770 with 8 GB of RAM.

Hybrid versus Right-looking (Linbox) and GPLU (time in s)

Matrix	Right-looking	GPLU	Hybrid
GL7d/GL7d24	34	276	11.6
Margulies/cat_ears_4_4	3	184	0.1
Homology/ch7-8.b4	173	0.2	0.2
Homology/ch7-8.b5	611	45	10.7

Results

Benchmark matrices: J.-G. Dumas Sparse Integer Matrix Collection (SIMC).
Experiment carried on an Intel Core i7-3770 with 8 GB of RAM.

Hybrid versus Right-looking (Linbox) and GPLU (time in s)

Matrix	Right-looking	GPLU	Hybrid
GL7d/GL7d24	34	276	11.6
Margulies/cat_ears_4_4	3	184	0.1
Homology/ch7-8.b4	173	0.2	0.2
Homology/ch7-8.b5	611	45	10.7

Hybrid versus Wiedmann (time in s)

Matrix	Wiedmann	Hybrid
M0,6-D7	20397	0.8
relat8	244	2
relat9	176694	2024

Conclusion

- Implemented in C and C++ in the SpaSM (SPArse System Modulo p) library and publicy available at:

`https://github.com/cbouilla/spasm`

Conclusion

- Implemented in C and C++ in the SpaSM (SPArse System Modulo p) library and publicy available at:

`https://github.com/cbouilla/spasm`

What's next ?

- Improve the research of the set of pivots.
- Combine with randomized algorithm
- More benchmark
  - More matrices from SIMC
  - On specific matrices collections (Cado-NFS).
- Contribute to the LinBox library.

Thank you for your time !