Computing the Rank Profile Matrix

Clément Pernet
joint work with Jean-Guillame Dumas and Ziad Sultan

Laboratoire de l'Informatique du Parallélisme,
Univ. Grenoble Alpes, Univ. de Lyon, CNRS, Inria.

AriC Seminar – October 8, 2015

Gaussian elimination in computer algebra

Swiss army knife for applications:

Matrix factorization **(LU decomposition)**

- Solving linear systems
- Computing determinants

Gaussian elimination in computer algebra

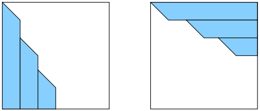
Swiss army knife for applications:

Matrix factorization **(LU decomposition)**

- Solving linear systems
- Computing determinants

Computing linear dependencies **(Echelon structure)**

- Basis of vector spaces (Krylov iteration)
- Echelon structure of the Macaulay matrix (Gröbner basis)



Rank profiles

Definition (Row Rank Profile: RowRP)

Given $A \in \mathsf{K}^{m\times n}, r = \mathsf{rank}(A)$.

informally: *first* r linearly independent rows

formally: lexico-minimal list of r indices of linearly independant rows.

Example

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rank profiles

Definition (Row Rank Profile: RowRP)

Given $A \in \mathsf{K}^{m \times n}, r = \mathsf{rank}(A)$.

informally: *first* r linearly independent rows

formally: lexico-minimal list of r indices of linearly independant rows.

Example

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rank = 3
RowRP = {1,2,4}

Rank profiles

Definition (Column Rank Profile: ColRP)

Given $A \in \mathsf{K}^{m\times n}, r = \mathsf{rank}(A)$.

informally: *first* r linearly independent columns

formally: lexico-minimal list of r linearly independant columns.

Example

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rank = 3
RowRP = {1,2,4}
ColRP = {1,2,3}

Rank profiles

Definition (Column Rank Profile: ColRP)

Given $A \in \mathsf{K}^{m \times n}, r = \mathsf{rank}(A)$.

informally: *first* r linearly independent columns

formally: lexico-minimal list of r linearly independant columns.

Example

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rank = 3
RowRP = {1,2,4}
ColRP = {1,2,3} $\rightarrow$ Generic ColRP.

Rank profiles

Definition (Column Rank Profile: ColRP)

Given $A \in \mathsf{K}^{m \times n}, r = \mathsf{rank}(A)$.

informally: *first* r linearly independent columns

formally: lexico-minimal list of r linearly independant columns.

Example

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rank = 3
RowRP = {1,2,4}
ColRP = {1,2,3} → Generic ColRP.

Generic Rank Profile: first r leading principal minors $\neq 0$

Generic rank profile $\not\Longleftarrow\Longrightarrow$ Generic Row RP and Genric ColRP

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

RowRP = ColRP = {1,2}

Rank profiles

Definition (Column Rank Profile: ColRP)

Given $A \in \mathsf{K}^{m \times n}, r = \mathsf{rank}(A)$.

informally: *first* r linearly independent columns

formally: lexico-minimal list of r linearly independant columns.

Example

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Rank = 3
RowRP = {1,2,4}
ColRP = {1,2,3} → Generic ColRP.

Generic Rank Profile: first r leading principal minors $\neq 0$

Generic rank profile $\not\Leftarrow\!\Rightarrow$ Generic Row RP and Genric ColRP

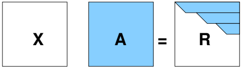
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

RowRP = ColRP = {1,2}
But $|A_{1,1}| = 0$

Relation to echelon forms

Transformation to echelon form

$\forall A\ \exists X$ non-singular s.t.

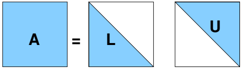


Relation to echelon forms:

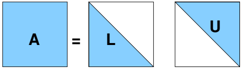
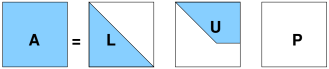
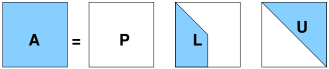
- ColRP unchanged by left multiplication with an invertible matrix

ColRP = pivot columns in the **row** echelon form

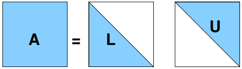
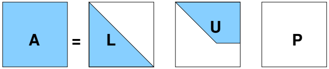
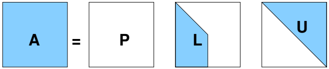
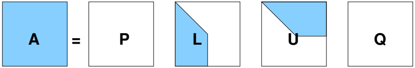
Triangular Matrix decompositions and rank profiles

Decomposition	Exists for	Unique
	Generic rank profile	Y

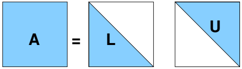
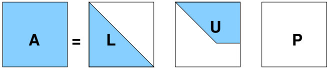
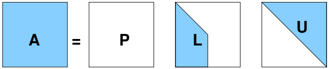
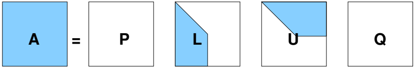
Triangular Matrix decompositions and rank profiles

Decomposition	Exists for	Unique
	Generic rank profile	Y
	Generic row rank profile	N
	Generic col rank profile	N

Triangular Matrix decompositions and rank profiles

Decomposition	Exists for	Unique
 A = L U	Generic rank profile	Y
 A = L U P	Generic row rank profile	N
 A = P L U	Generic col rank profile	N
 A = P L U Q	Any matrix	N

Triangular Matrix decompositions and rank profiles

Decomposition	Exists for	Unique
	Generic rank profile	Y
	Generic row rank profile	N
	Generic col rank profile	N
	Any matrix	N

$\rightarrow P, Q$ may reveal row and/or col rank profiles.

Computing rank profiles

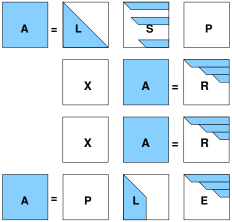
Via Gaussian elimination revealing row echelon forms:

[Ibarra, Moran and Hui 82]

[Keller-Gehrig 85]

[Storjohann 00]

[Jeannerod, P. and Storjohann 13]



Computing rank profiles

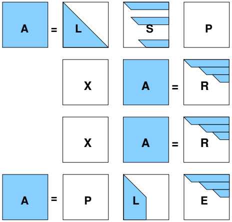
Via Gaussian elimination revealing row echelon forms:

[Ibarra, Moran and Hui 82]

[Keller-Gehrig 85]

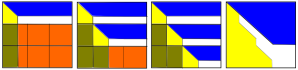
[Storjohann 00]

[Jeannerod, P. and Storjohann 13]



Lessons learned (or what we thought was necessary):

- treat rows in order
- exhaust all columns before considering the next row
- **slab** block splitting (recursive or iterative)



⇝ similar to partial pivoting

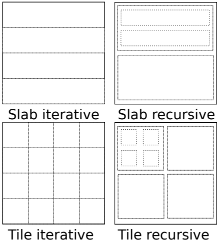
Motivation

Need more flexible blocking

Slab blocking

- leads to inefficient memory access patterns
- is harder to parallelize

Tile blocking instead ?



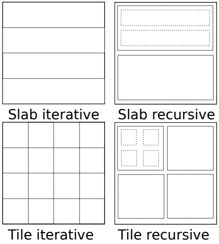
Motivation

Need more flexible blocking

Slab blocking

- leads to inefficient memory access patterns
- is harder to parallelize

Tile blocking instead ?



Gathering linear independence invariants

Two ways to look at a matrix (looking left or right):

- Row rank profile, column echelon form
- Column rank profile, row echelon form

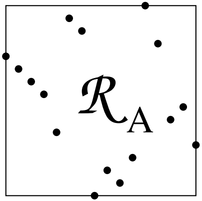
Unique invariant?

Outline

The rank profile Matrix

Theorem

Let $A \in \mathrm{F}^{m \times n}$.
There exists a unique, $m \times n$, *rank*(A)*-sub-permutation matrix* $\mathcal{R}^A$ *of which every leading sub-matrix has the same rank as the corresponding leading sub-matrix of* A.



The rank profile Matrix

Theorem

Let $A \in \mathrm{F}^{m\times n}$.
There exists a unique, $m \times n$, *rank*(A)*-sub-permutation matrix* $\mathcal{R}^A$ *of which every leading sub-matrix has the same rank as the corresponding leading sub-matrix of* A.

Example

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 7 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rank profile Matrix

Theorem

Let $A \in \mathrm{F}^{m\times n}$.
There exists a unique, $m \times n$, rank(A)-sub-permutation matrix $\mathcal{R}^A$ of which every leading sub-matrix has the same rank as the corresponding leading sub-matrix of A.

Example

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 7 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rank profile Matrix

Theorem

Let $A \in \mathrm{F}^{m \times n}$.
There exists a unique, $m \times n$, *rank*(A)*-sub-permutation matrix* $\mathcal{R}^A$ *of which every leading sub-matrix has the same rank as the corresponding leading sub-matrix of* A.

Example

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 7 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rank profile Matrix

Theorem

Let $A \in \mathrm{F}^{m\times n}$.
There exists a unique, $m \times n$, rank(A)-sub-permutation matrix $\mathcal{R}^A$ of which every leading sub-matrix has the same rank as the corresponding leading sub-matrix of A.

Example

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 7 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Properties of the rank profile matrix

Properties

- A *invertible* $\Rightarrow \mathcal{R}^A$ *is a permutation matrix*
- A *is square with generic rank profile* $\Rightarrow \mathcal{R}^A = I_n$
- *RowRP*$(A) =$ *RowSupport*$(\mathcal{R}^A)$
- *ColRP*$(A) =$ *ColSupport*$(\mathcal{R}^A)$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 7 \end{bmatrix} \qquad \begin{matrix} \text{RowRP} = \{1,3,4\} \\ \text{ColRP} = \{1,2,4\} \end{matrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Properties of the rank profile matrix

Properties

- A *invertible* $\Rightarrow \mathcal{R}^A$ *is a permutation matrix*
- A *is square with generic rank profile* $\Rightarrow \mathcal{R}^A = I_n$
- *RowRP*(A) = *RowSupport*$(\mathcal{R}^A)$
- *ColRP*(A) = *ColSupport*$(\mathcal{R}^A)$
- *RowRP*$(A_{1..i,1..j})$ = *RowSupport*$({\mathcal{R}^A}_{1..i,1..j})$
- *ColRP*$(A_{1..i,1..j})$ = *ColSupport*$({\mathcal{R}^A}_{1..i,1..j})$

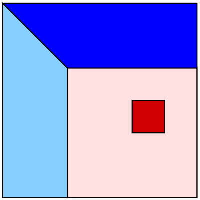
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 2 & 5 & 4 & 7 \end{bmatrix}$$

$$\text{RowRP} = \{1,3\}$$
$$\text{ColRP} = \{1,2\}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Outline

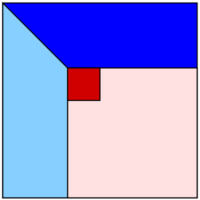
Anatomy of a PLUQ decomposition



Four types of elementary operations:

Search: finding a pivot

Anatomy of a PLUQ decomposition

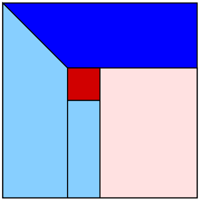


Four types of elementary operations:

Search: finding a pivot

Permutation: moving the pivot to the main diagonal

Anatomy of a PLUQ decomposition



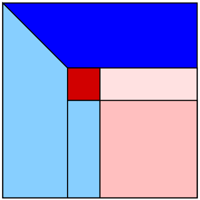
Four types of elementary operations:

Search: finding a pivot

Permutation: moving the pivot to the main diagonal

Normalization: computing L: $l_{i,k} \leftarrow \frac{a_{i,k}}{a_{k,k}}$

Anatomy of a PLUQ decomposition



Four types of elementary operations:

Search: finding a pivot

Permutation: moving the pivot to the main diagonal

Normalization: computing L: $l_{i,k} \leftarrow \frac{a_{i,k}}{a_{k,k}}$

Update: applying the elimination $a_{i,j} \leftarrow a_{i,j} - \frac{a_{i,k}a_{k,j}}{a_{k,k}}$

Impact on the PLUQ decomposition

Normalization: determines whether L or U is unit diagonal

Impact on the PLUQ decomposition

Normalization: determines whether L or U is unit diagonal

Update: no impact on the decomposition, only in the scheduling:

- iterative, tile/slab iterative, recursive,
- left/right looking, Crout

Impact on the PLUQ decomposition

Normalization: determines whether L or U is unit diagonal

Update: no impact on the decomposition, only in the scheduling:

- iterative, tile/slab iterative, recursive,
- left/right looking, Crout

Search: defines the first r values of P and Q

Impact on the PLUQ decomposition

Normalization: determines whether L or U is unit diagonal

Update: no impact on the decomposition, only in the scheduling:

- iterative, tile/slab iterative, recursive,
- left/right looking, Crout

Search: defines the first r values of P and Q

Permutation: impacts all values of P and Q

Impact on the PLUQ decomposition

Normalization: determines whether L or U is unit diagonal

Update: no impact on the decomposition, only in the scheduling:

- iterative, tile/slab iterative, recursive,
- left/right looking, Crout

Search: defines the first r values of P and Q

Permutation: impacts all values of P and Q

Problem (Reformulation)

Under what conditions on the **Search** *and* **Permutation** *operations does a PLUQ decomposition algorithm reveals RowRP, ColRP or* $\mathcal{R}^A$*?*

The Pivoting matrix

Definition (The pivoting matrix)

Given a PLUQ decomposition $A = PLUQ$ with rank r, define

$$\Pi_{P,Q} = P \begin{bmatrix} I_r & \\ & \end{bmatrix} Q.$$

Locates the position of the pivots in the matrix A.

The Pivoting matrix

Definition (The pivoting matrix)

Given a PLUQ decomposition $A = PLUQ$ with rank r, define

$$\Pi_{P,Q} = P \begin{bmatrix} I_r & \\ & \end{bmatrix} Q.$$

Locates the position of the pivots in the matrix A.

Problem (Rank profile revealing PLUQ decompositions)

Under which conditions

- $\Pi_{P,Q} = \mathcal{R}^A$

The Pivoting matrix

Definition (The pivoting matrix)

Given a PLUQ decomposition $A = PLUQ$ with rank r, define

$$\Pi_{P,Q} = P \begin{bmatrix} I_r & \\ & \end{bmatrix} Q.$$

Locates the position of the pivots in the matrix A.

Problem (Rank profile revealing PLUQ decompositions)

Under which conditions

- $\Pi_{P,Q} = \mathcal{R}^A$
- $RowSupp(\Pi_{P,Q}) = RowSupp(\mathcal{R}^A) = RowRP(A)$ *(Weaker)*
- $ColSupp(\Pi_{P,Q}) = ColSupp(\mathcal{R}^A) = ColRP(A)$ *(Weaker)*

The Search operation

Various strategies depending on the context

Numerical stability: find the absolute largest pivot

Data locality: find pivot not too far from the main diagonal

Sparsity: find pivot that minimizes/reduce fill-in

The Search operation

Various strategies depending on the context

Numerical stability: find the absolute largest pivot

Data locality: find pivot not too far from the main diagonal

Sparsity: find pivot that minimizes/reduce fill-in

Search revealing rank profiles

- No stability issue over exact domains
- Intuition: must **minimize** some **ordering of the row/col indices** (notion of rank profile)

The Search operation

Various strategies depending on the context

Numerical stability: find the absolute largest pivot

Data locality: find pivot not too far from the main diagonal

Sparsity: find pivot that minimizes/reduce fill-in

Search revealing rank profiles

- No stability issue over exact domains
- Intuition: must **minimize** some **ordering of the row/col indices** (notion of rank profile)

Example

Search: "Any non zero element on the topmost row":

$$A = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \Rightarrow \mathcal{R}^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

The Search operation

Various strategies depending on the context

Numerical stability: find the absolute largest pivot

Data locality: find pivot not too far from the main diagonal

Sparsity: find pivot that minimizes/reduce fill-in

Search revealing rank profiles

- No stability issue over exact domains
- Intuition: must **minimize** some **ordering of the row/col indices** (notion of rank profile)

Example

Search: "Any non zero element on the topmost row":

$$A = \begin{bmatrix} 2&0&3&0\\1&0&0&0\\0&0&4&0\\0&2&0&1 \end{bmatrix} \Rightarrow \mathcal{R}^A = \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&0&0&0\\0&1&0&0 \end{bmatrix}, \quad \Pi_{P,Q} = \begin{bmatrix} 0&0&1&0\\1&0&0&0\\0&0&0&0\\0&1&0&0 \end{bmatrix}.$$

The Search operation

Various strategies depending on the context

Numerical stability: find the absolute largest pivot

Data locality: find pivot not too far from the main diagonal

Sparsity: find pivot that minimizes/reduce fill-in

Search revealing rank profiles

- No stability issue over exact domains
- Intuition: must **minimize** some **ordering of the row/col indices** (notion of rank profile)

Example

Search: "Any non zero element on the topmost row":

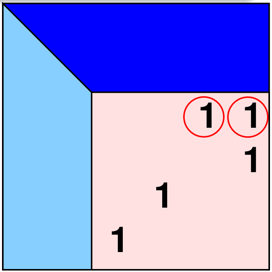
$$A = \begin{bmatrix} 2&0&3&0\\ 1&0&0&0\\ 0&0&4&0\\ 0&2&0&1 \end{bmatrix} \Rightarrow \mathcal{R}^A = \begin{bmatrix} 1&0&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&1&0&0 \end{bmatrix}, \quad \Pi_{P,Q} = \begin{bmatrix} 0&0&1&0\\ 1&0&0&0\\ 0&0&0&0\\ 0&1&0&0 \end{bmatrix}. \qquad \rightsquigarrow \text{RowRP}=\{1,2,4\}$$

Pivoting and permutation strategies

Pivot Search

Pivot's (i, j) position minimizes some pre-order:

Row order: any non-zero on the first non-zero row

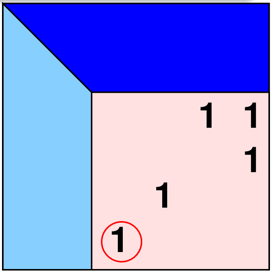


Pivoting and permutation strategies

Pivot Search

Pivot's (i, j) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col



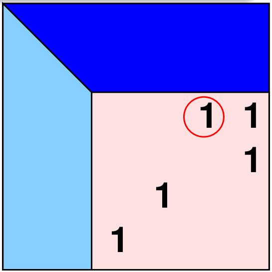
Pivoting and permutation strategies

Pivot Search

Pivot's (i, j) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col

Lex order: first non-zero on the first non-zero row



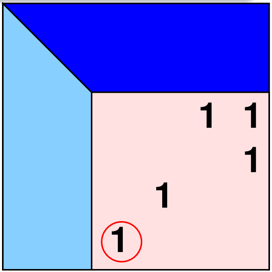
Pivoting and permutation strategies

Pivot Search

Pivot's (i, j) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col

Lex/RevLex order: first non-zero on the first non-zero row/col



Pivoting and permutation strategies

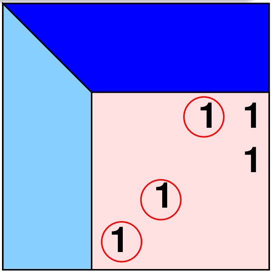
Pivot Search

Pivot's (i,j) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col

Lex/RevLex order: first non-zero on the first non-zero row/col

Product order: first non-zero in the (i,j) leading sub-matrix



Sufficient ?

Is lexicographic ordering sufficient to reveal both rank profiles?

Example

With a lexicographic ordering

1. $$A = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \Rightarrow \mathcal{R}^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \Pi_{P,Q}$$

Sufficient ?

Is lexicographic ordering sufficient to reveal both rank profiles?

Example

With a lexicographic ordering

1. $A = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \Rightarrow \mathcal{R}^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \Pi_{P,Q}$
2. But $A = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix} \rightsquigarrow \mathcal{R}^A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ and $\Pi_{P,Q} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$.

Sufficient ?

Is lexicographic ordering sufficient to reveal both rank profiles?

Example

With a lexicographic ordering

1. $A = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \Rightarrow \mathcal{R}^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \Pi_{P,Q}$
2. But $A = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix} \rightsquigarrow \mathcal{R}^A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ and $\Pi_{P,Q} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$.

$\rightsquigarrow$ Pivot Swaps mix-up precedence between rows/cols.

$\rightsquigarrow$ **Permutations** also have to be considered

Pivoting and permutation strategies

Pivot Search

Pivot's (i, j) position minimizes some pre-order:

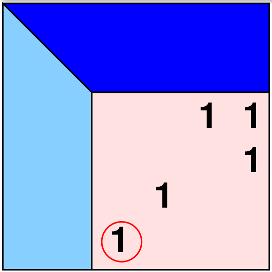
Row/Col order: any non-zero on the first non-zero row/col

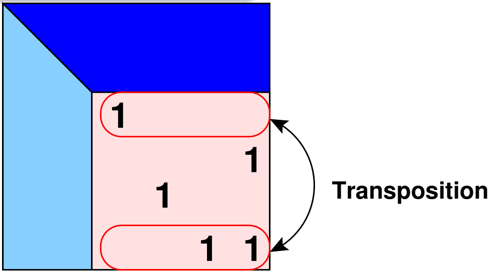
Lex/RevLex order: first non-zero on the first non-zero row/col

Product order: first non-zero in the (i, j) leading sub-matrix

Permutation

- Transpositions





Pivoting and permutation strategies

Pivot Search

Pivot's (i, j) position minimizes some pre-order:

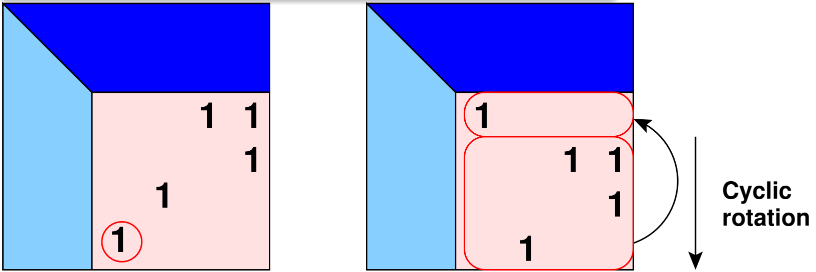
Row/Col order: any non-zero on the first non-zero row/col

Lex/RevLex order: first non-zero on the first non-zero row/col

Product order: first non-zero in the (i, j) leading sub-matrix

Permutation

- Transpositions
- Cyclic Rotations



Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

Search	**Row perm.**	**Col. perm.**	RowRP	ColRP	$\mathcal{R}^A$	**Instance**
Row order Col. order						
Lexico.						
Rev. lex.						
Product						

Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

Search	**Row perm.**	**Col. perm.**	RowRP	ColRP	$\mathcal{R}^A$	**Instance**
Row order Col. order	Transposition	Transposition				[IMH82] [JPS13]
Lexico.						
Rev. lex.						
Product						

- RowRP $= \left[\, 1\ 2\ \dots\ m \,\right] P \left[\begin{smallmatrix} I_r \\ 0 \end{smallmatrix}\right]$

Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

Search	Row perm.	Col. perm.	RowRP	ColRP	$\mathcal{R}^A$	Instance
Row order	Transposition	Transposition				[IMH82] [JPS13]
Col. order	Transposition	Transposition				[KG85] [JPS13]
Lexico.						
Rev. lex.						
Product						

- RowRP $= \left[\,1\ 2\ \ldots\ m\,\right] P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- ColRP $= \left[\,I_r\ 0\,\right] Q \left[\,1\ 2\ \ldots\ m\,\right]^T$

Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

Search	**Row perm.**	**Col. perm.**	RowRP	ColRP	$\mathcal{R}^A$	**Instance**
Row order	Transposition	Transposition				[IMH82] [JPS13]
Col. order	Transposition	Transposition				[KG85] [JPS13]
Lexico.	Transposition	Transposition				[Sto00]
Rev. lex.	Transposition	Transposition				[Sto00]
Product						

- RowRP $= [\,1\ 2\ \ldots\ m\,]\, P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- ColRP $= [\,I_r\ 0\,]\, Q\, [\,1\ 2\ \ldots\ m\,]^T$

Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

Search	Row perm.	Col. perm.	RowRP	ColRP	$\mathcal{R}^A$	Instance
Row order	Transposition	Transposition				[IMH82] [JPS13]
Col. order	Transposition	Transposition				[KG85] [JPS13]
Lexico.	Transposition	Transposition				[Sto00]
Rev. lex.	Transposition	Transposition				[Sto00]
Product	Rotation	Rotation				[DPS13]

- RowRP $= \left[\, 1\ 2\ \dots\ m \,\right] P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- ColRP $= \left[\, I_r\ 0 \,\right] Q \left[\, 1\ 2\ \dots\ m \,\right]^T$
- $\mathcal{R}^A = P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} Q$

Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

Search	Row perm.	Col. perm.	RowRP	ColRP	$\mathcal{R}^A$	Instance
Row order	Transposition	Transposition				[IMH82] [JPS13]
Col. order	Transposition	Transposition				[KG85] [JPS13]
Lexico.	Transposition	Transposition				[Sto00]
Rev. lex.	Transposition	Transposition				[Sto00]
Product	Rotation	Transposition				[DPS15]
Product	Transposition	Rotation				[DPS15]
Product	Rotation	Rotation				[DPS13]

- RowRP $= \begin{bmatrix} 1 & 2 & \dots & m \end{bmatrix} P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- ColRP $= \begin{bmatrix} I_r & 0 \end{bmatrix} Q \begin{bmatrix} 1 & 2 & \dots & m \end{bmatrix}^T$
- $\mathcal{R}^A = P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} Q$

Pivoting strategies revealing rank profiles

For any type of PLUQ algorithm: iterative / block iterative / recursive

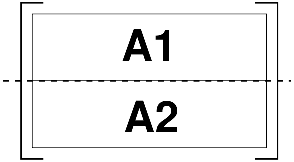
Search	Row perm.	Col. perm.	RowRP	ColRP	$\mathcal{R}^A$	Instance
Row order	Transposition	Transposition				[IMH82] [JPS13]
Col. order	Transposition	Transposition				[KG85] [JPS13]
Lexico.	Transposition	Transposition				[Sto00]
Lexico.	Transposition	Rotation	✓	✓	✓	[DPS15]
Lexico.	Rotation	Rotation	✓	✓	✓	[DPS15]
Rev. lex.	Transposition	Transposition				[Sto00]
Rev. lex.	Rotation	Transposition	✓	✓	✓	[DPS15]
Rev. lex.	Rotation	Rotation	✓	✓	✓	[DPS15]
Product	Rotation	Transposition				[DPS15]
Product	Transposition	Rotation				[DPS15]
Product	Rotation	Rotation				[DPS13]

- RowRP $= \begin{bmatrix} 1 & 2 & \dots & m \end{bmatrix} P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- ColRP $= \begin{bmatrix} I_r & 0 \end{bmatrix} Q \begin{bmatrix} 1 & 2 & \dots & m \end{bmatrix}^T$
- $\mathcal{R}^A = P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} Q$

Outline

The slab recursive algorithm

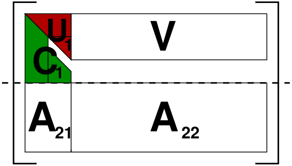
Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



1. Split A Row-wise

The slab recursive algorithm

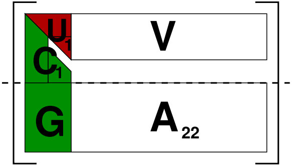
Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



1. Split A Row-wise
2. Recursive call on A_1

The slab recursive algorithm

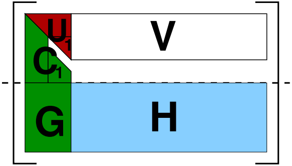
Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



1. Split A Row-wise
2. Recursive call on A_1
3. $G \leftarrow A_{21}U_1^{-1}$ (`trsm`)

The slab recursive algorithm

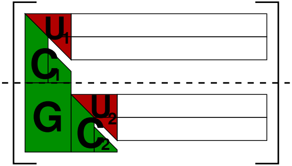
Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



1. Split A Row-wise
2. Recursive call on A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (`trsm`)
4. $H \leftarrow A_{22} - G \times V$ (`MM`)

The slab recursive algorithm

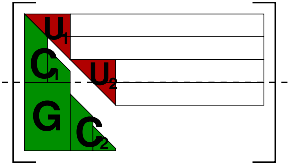
Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



1. Split A Row-wise
2. Recursive call on A_1
3. $G \leftarrow A_{21}U_1^{-1}$ (`trsm`)
4. $H \leftarrow A_{22} - G \times V$ (`MM`)
5. Recursive call on H

The slab recursive algorithm

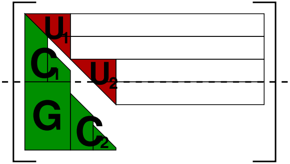
Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



1. Split A Row-wise
2. Recursive call on A_1
3. $G \leftarrow A_{21}U_1^{-1}$ (trsm)
4. $H \leftarrow A_{22} - G \times V$ (MM)
5. Recursive call on H
6. Row permutations

The slab recursive algorithm

Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



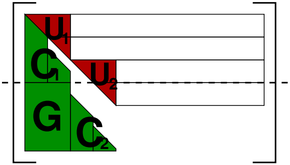
1. Split A Row-wise
2. Recursive call on A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (trsm)
4. $H \leftarrow A_{22} - G \times V$ (MM)
5. Recursive call on H
6. Row permutations

Implements the lexicographic order search.

- Col/Row Transpositions : Computes the ColRP

The slab recursive algorithm

Slab Recursive LU [IMH82, KG85, Sto00, JPS13]



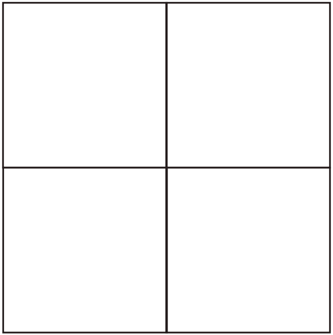
1. Split A Row-wise
2. Recursive call on A_1
3. $G \leftarrow A_{21} U_1^{-1}$ (trsm)
4. $H \leftarrow A_{22} - G \times V$ (MM)
5. Recursive call on H
6. Row permutations

Implements the lexicographic order search.

- Col/Row Transpositions : Computes the ColRP
- Row Rotations : Computes $\mathcal{R}^A$ [DPS15]

The tiled recursive algorithm

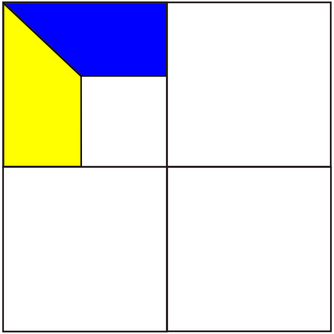
Dumas, P. and Sultan 13



2×2 block splitting

The tiled recursive algorithm

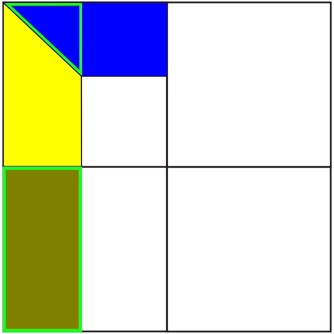
Dumas, P. and Sultan 13



Recursive call

The tiled recursive algorithm

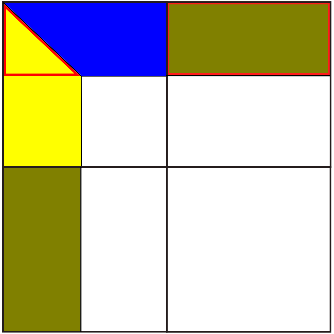
Dumas, P. and Sultan 13



TRSM: $B \leftarrow BU^{-1}$

The tiled recursive algorithm

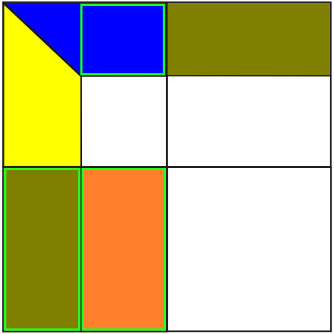
Dumas, P. and Sultan 13



TRSM: $B \leftarrow L^{-1}B$

The tiled recursive algorithm

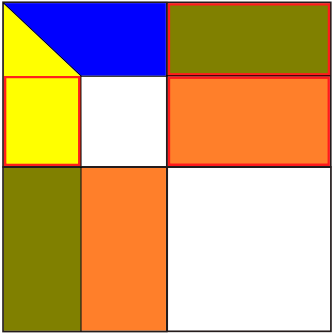
Dumas, P. and Sultan 13



MatMul: $C \leftarrow C - A \times B$

The tiled recursive algorithm

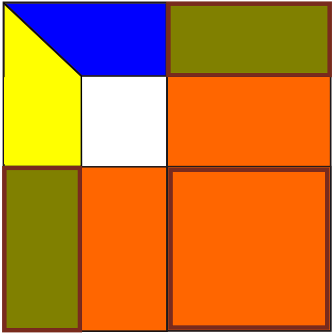
Dumas, P. and Sultan 13



MatMul: $C \leftarrow C - A \times B$

The tiled recursive algorithm

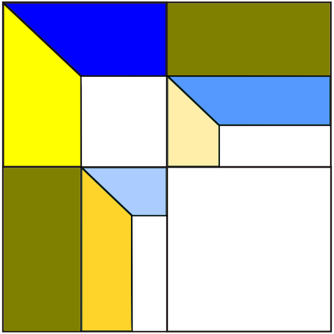
Dumas, P. and Sultan 13



`MatMul`: $C \leftarrow C - A \times B$

The tiled recursive algorithm

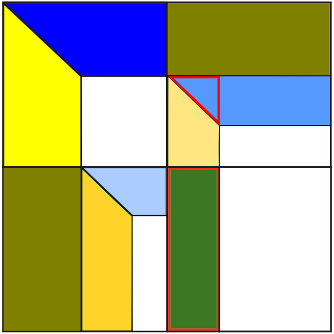
Dumas, P. and Sultan 13



2 independent recursive calls (compatible with the **product order**)

The tiled recursive algorithm

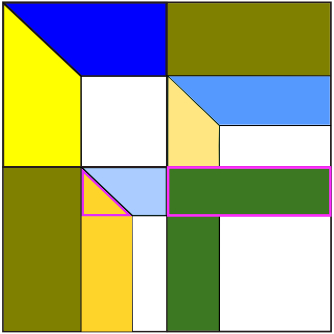
Dumas, P. and Sultan 13



TRSM: $B \leftarrow BU^{-1}$

The tiled recursive algorithm

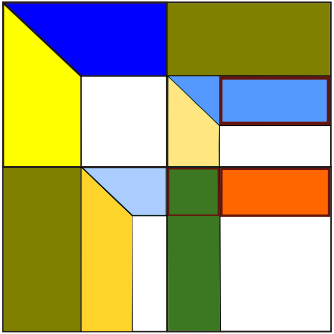
Dumas, P. and Sultan 13



TRSM: $B \leftarrow L^{-1}B$

The tiled recursive algorithm

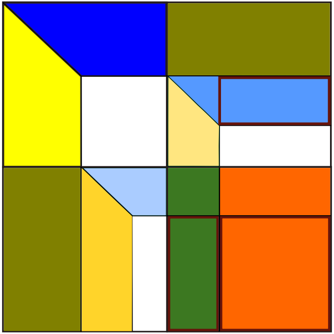
Dumas, P. and Sultan 13



MatMul: $C \leftarrow C - A \times B$

The tiled recursive algorithm

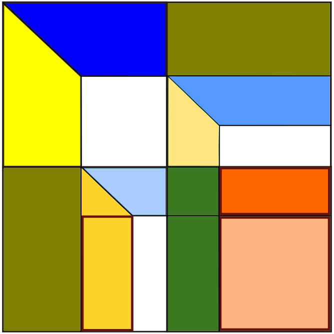
Dumas, P. and Sultan 13



MatMul: $C \leftarrow C - A \times B$

The tiled recursive algorithm

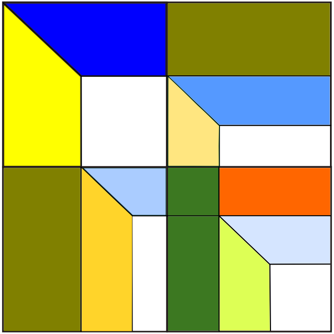
Dumas, P. and Sultan 13



MatMul: $C \leftarrow C - A \times B$

The tiled recursive algorithm

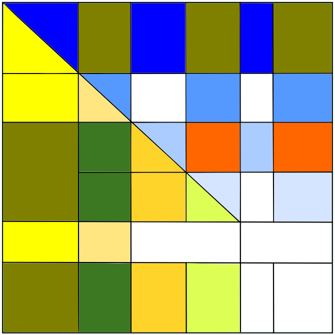
Dumas, P. and Sultan 13



Recursive call

The tiled recursive algorithm

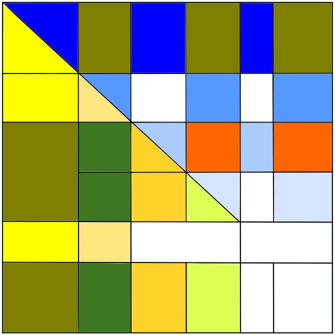
Dumas, P. and Sultan 13



Puzzle game (block **rotations**)

The tiled recursive algorithm

Dumas, P. and Sultan 13



- $O(mnr^{\omega-2})$ ($2/3n^3$ for $\omega = 3$)
- fewer modular reductions than slab algorithms
- rank deficiency introduces parallelism

Iterative algorithms

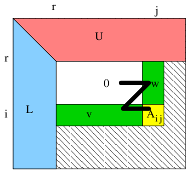
- Unefficient with large problems
- Good for base case implementations (faster in-cache computation)

Iterative algorithms

- Unefficient with large problems
- Good for base case implementations (faster in-cache computation)

Which base case algorithm?

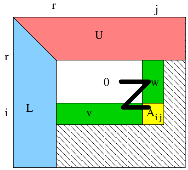
- Formerly [DPS13]: **product order** iterative algorithm
  - ✗ many permutations
  - ✗ many modular reductions



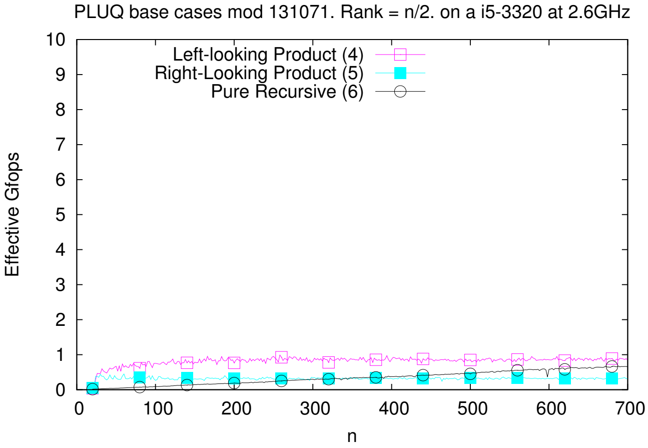
Iterative algorithms

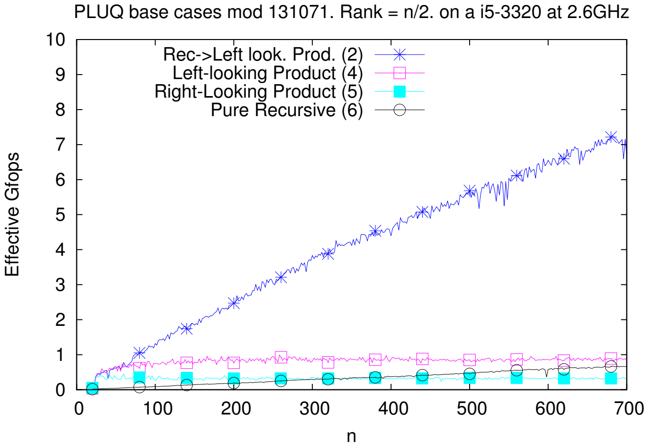
- Unefficient with large problems
- Good for base case implementations (faster in-cache computation)

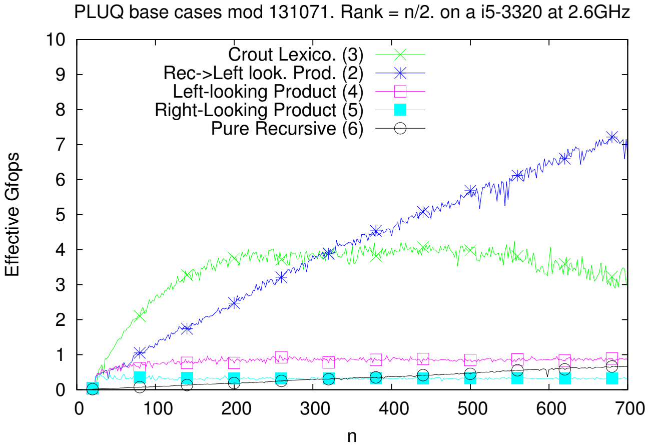
Which base case algorithm?

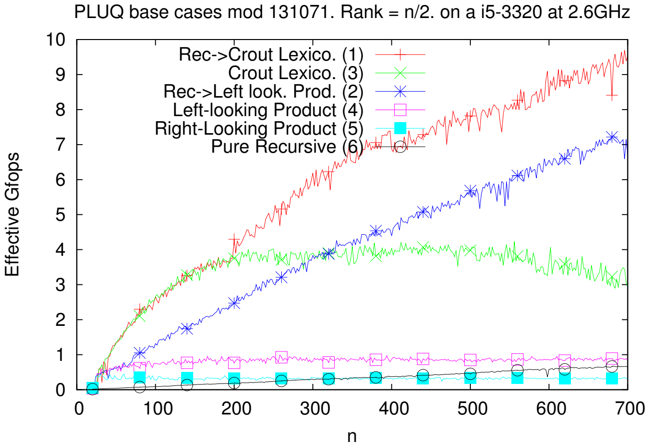


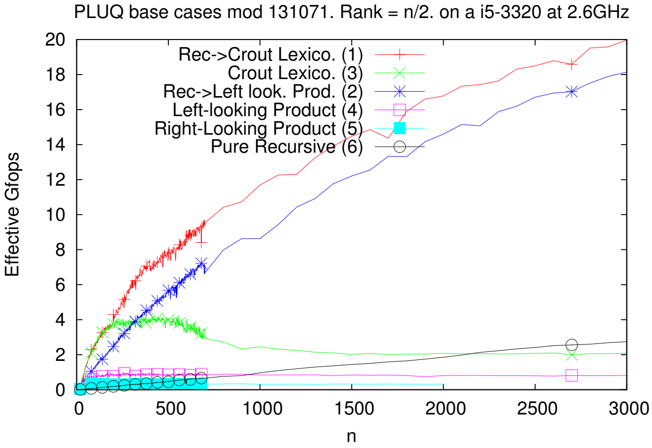
- Formerly [DPS13]: **product order** iterative algorithm
  - ✗ many permutations
  - ✗ many modular reductions
- [DPS15]: Simply use the schoolbook algorithm (Lexico+Rotations)
  - ✓ fewer permutations
  - ✓ modular reductions delayed more easily
  - ✓ Crout variant: better data access pattern

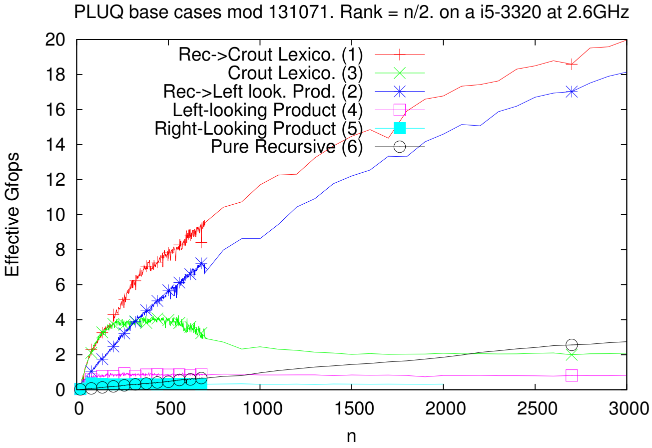

PLUQ base cases mod 131071. Rank = n/2. on a i5-3320 at 2.6GHz
Left-looking Product (4)
Right-Looking Product (5)
Pure Recursive (6)
Effective Gfops
n
0
100
200
300
400
500
600
700
0
1
2
3
4
5
6
7
8
9
10


PLUQ base cases mod 131071. Rank = n/2. on a i5-3320 at 2.6GHz
Rec->Left look. Prod. (2)
Left-looking Product (4)
Right-Looking Product (5)
Pure Recursive (6)
Effective Gfops
n
0
1
2
3
4
5
6
7
8
9
10
0
100
200
300
400
500
600
700


PLUQ base cases mod 131071. Rank = n/2. on a i5-3320 at 2.6GHz
Crout Lexico. (3)
Rec->Left look. Prod. (2)
Left-looking Product (4)
Right-Looking Product (5)
Pure Recursive (6)
Effective Gfops
n
0
1
2
3
4
5
6
7
8
9
10
0
100
200
300
400
500
600
700


PLUQ base cases mod 131071. Rank = n/2. on a i5-3320 at 2.6GHz
Rec->Crout Lexico. (1)
Crout Lexico. (3)
Rec->Left look. Prod. (2)
Left-looking Product (4)
Right-Looking Product (5)
Pure Recursive (6)
Effective Gfops
n
0
1
2
3
4
5
6
7
8
9
10
100
200
300
400
500
600
700


PLUQ base cases mod 131071. Rank = n/2. on a i5-3320 at 2.6GHz
Rec->Crout Lexico. (1)
Crout Lexico. (3)
Rec->Left look. Prod. (2)
Left-looking Product (4)
Right-Looking Product (5)
Pure Recursive (6)
Effective Gfops
n
0
2
4
6
8
10
12
14
16
18
20
0
500
1000
1500
2000
2500
3000



- > 2 Gfops improvement
- Implemented in `FFLAS-FFPACK` (kernel of `LinBox`).

Outline

Malaschonok LEU decomposition

[Malaschonok'10]: $A = L \cdot E \cdot U$

- E is an r-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^{\omega}-4} MM(m, n)$ with $m = n = 2^k$.
- no connection to rank profile nor echelon form
- no rank sensitive complexity

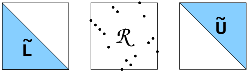
Malaschonok LEU decomposition

[Malaschonok'10]: $A = L \cdot E \cdot U$

- E is an r-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^{\omega}-4} MM(m,n)$ with $m = n = 2^k$.
- no connection to rank profile nor echelon form
- no rank sensitive complexity

Fact

$$E = \mathcal{R}^A$$



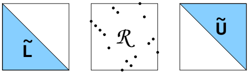
Malaschonok LEU decomposition

[Malaschonok'10]: $A = L \cdot E \cdot U$

- E is an r-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^{\omega}-4} MM(m,n)$ with $m = n = 2^k$.
- no connection to rank profile nor echelon form
- no rank sensitive complexity

Fact

$$E = \mathcal{R}^A$$



$$A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} \begin{bmatrix} U & V \\ & I_{n-r} \end{bmatrix} Q$$

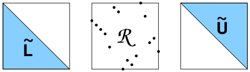
Malaschonok LEU decomposition

[Malaschonok'10]: $A = L \cdot E \cdot U$

- E is an r-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^\omega - 4} MM(m,n)$ with $m = n = 2^k$.
- no connection to rank profile nor echelon form
- no rank sensitive complexity

Fact

$$E = \mathcal{R}^A$$



$$A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} Q Q^T \begin{bmatrix} U & V \\ & I_{n-r} \end{bmatrix} Q$$

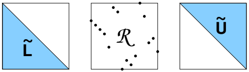
Malaschonok LEU decomposition

[Malaschonok'10]: $A = L \cdot E \cdot U$

- E is an r-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^{\omega}-4} MM(m,n)$ with $m = n = 2^k$.
- no connection to rank profile nor echelon form
- no rank sensitive complexity

Fact

$$E = \mathcal{R}^A$$



$$A = PLUQ = \underbrace{P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T}_{\overline{L}} \underbrace{P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} Q}_{\Pi_{P,Q}} \underbrace{Q^T \begin{bmatrix} U & V \\ & I_{n-r} \end{bmatrix} Q}_{\overline{U}}$$

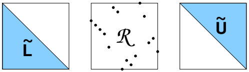
Malaschonok LEU decomposition

[Malaschonok'10]: $A = L \cdot E \cdot U$

- E is an r-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^\omega - 4} MM(m,n)$ with $m = n = 2^k$.
- no connection to rank profile nor echelon form
- no rank sensitive complexity

Fact

$$E = \mathcal{R}^A$$



$$A = PLUQ = \underbrace{P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T}_{\overline{L}} \underbrace{P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} Q}_{\Pi_{P,Q}} \underbrace{Q^T \begin{bmatrix} U & V \\ & I_{n-r} \end{bmatrix} Q}_{\overline{U}}$$

With appropriate pivoting: $\Pi_{P,Q} = \mathcal{R}(A)$

LUP and PLU decompositions

LUP

If A has generic RowRP

- $LUP(A)$ with Lex order and col. rot.: $\rightsquigarrow \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} P = \mathcal{R}^A$

In particular, if A has full row rank and $m = n$: $\rightsquigarrow P = \mathcal{R}^A$

LUP and PLU decompositions

LUP

If A has generic RowRP

- $LUP(A)$ with Lex order and col. rot.: $\rightsquigarrow \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} P = \mathcal{R}^A$

In particular, if A has full row rank and $m = n$: $\rightsquigarrow P = \mathcal{R}^A$

PLU

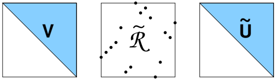
If A has generic ColRP

- $PLU(A)$ with RevLex order and row rot. $\rightsquigarrow P \begin{bmatrix} I_r & \\ & 0 \end{bmatrix} = \mathcal{R}^A$

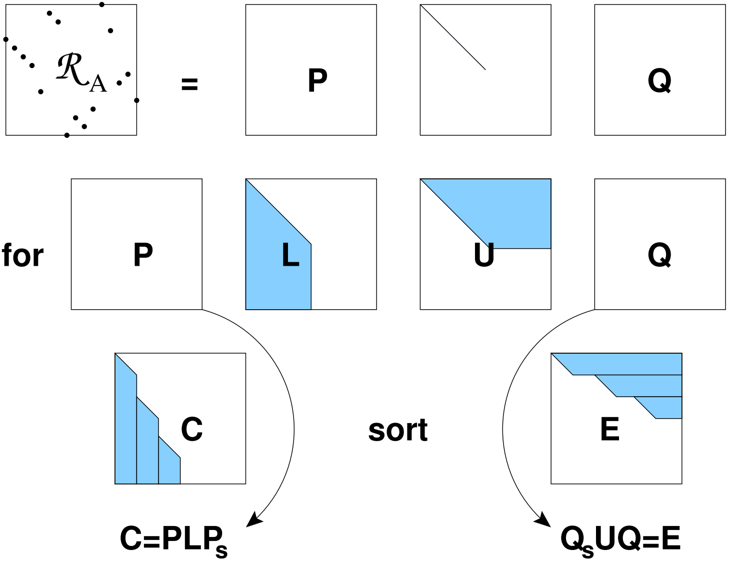
In particular, if A has full column rank and $m = n$: $\rightsquigarrow P = \mathcal{R}^A$

Bruhat decomposition

- If $A = \tilde{L}\mathcal{R}^A\tilde{U}$, then
  - For J_n the unit anti-diagonal matrix,
  - $V = J_n\tilde{L}J_n$ is upper triangular
  - $\tilde{\mathcal{R}} = J_n\mathcal{R}^A$ is a rank r sub-permutation
  - $A = V\tilde{\mathcal{R}}\tilde{U}$ (Bruhat decomposition)



Echelon forms



Outline

1. The rank profile Matrix
2. Computing the rank profile matrix
3. Algorithmic instances
4. Relations to other decompositions
5. The small rank case

Small rank

When $r \ll m, n$, $O(mnr^{\omega-2})$ can be too expensive.
(Compressed sensing applications)

[Cheung Kwok Lau'12]: Compute the rank r and r linearly independent rows in $\tilde{O}(r^\omega + mn)$ probabilistic

Small rank

When $r \ll m, n$, $O(mnr^{\omega-2})$ can be too expensive.
(Compressed sensing applications)

[Cheung Kwok Lau'12]: Compute the rank r and r linearly independent rows in $\tilde{O}(r^\omega + mn)$ probabilistic

[Storjohann Yang'14:] Rank profile in $\tilde{O}(r^3 + mn)$ probabilistic.

Small rank

When $r \ll m, n$, $O(mnr^{\omega-2})$ can be too expensive.
(Compressed sensing applications)

[Cheung Kwok Lau'12]: Compute the rank r and r linearly independent rows in $\tilde{O}(r^{\omega} + mn)$ probabilistic

[Storjohann Yang'14:] Rank profile in $\tilde{O}(r^3 + mn)$ probabilistic.

[Storjohann Yang'15:] Rank profile in $\tilde{O}(r^{\omega} + mn)$ probabilistic.

Small rank

When $r \ll m, n$, $O(mnr^{\omega-2})$ can be too expensive.
(Compressed sensing applications)

[Cheung Kwok Lau'12]: Compute the rank r and r linearly independent rows in $\tilde{O}(r^\omega + mn)$ probabilistic

[Storjohann Yang'14:] Rank profile in $\tilde{O}(r^3 + mn)$ probabilistic.

[Storjohann Yang'15:] Rank profile in $\tilde{O}(r^\omega + mn)$ probabilistic.

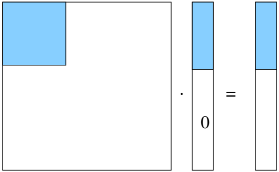
Can the rank profile matrix be computed in such complexities?

[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.



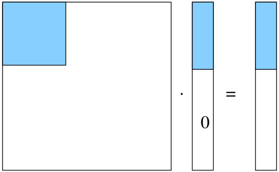
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(sn)$



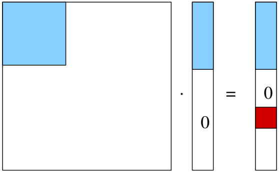
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(sn)$



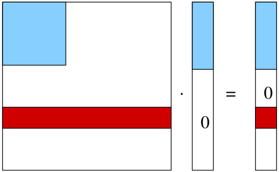
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(sn)$



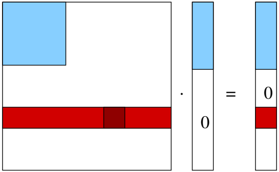
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(sn)$



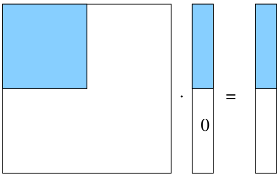
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(sn)$
2. Compute A_{s+1}^{-1} by rank 1 updates $\rightsquigarrow O(s^2)$



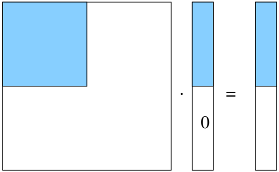
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(sn)$
2. Compute A_{s+1}^{-1} by rank 1 updates $\rightsquigarrow O(s^2)$



- Use the vector b to compress row linear dependency information

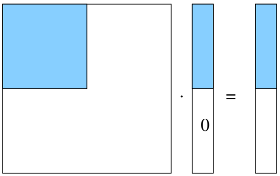
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(s \log n)$
2. Compute A_{s+1}^{-1} by rank 1 updates $\rightsquigarrow O(s^2)$



- Use the vector b to compress row linear dependency information
- Improved by linear independence oracles

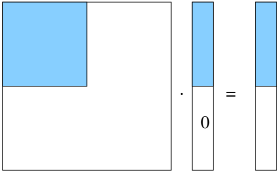
[Storjohann Yang'14] Linear System Oracle

Sketch of the $\tilde{O}(r^3 + mn)$ algorithm

Incrementally for $s = 1..\text{rank}(A)$, maintain

- an $s \times s$ invertible sub-matrix A_s of A.
- its inverse A_s^{-1}
- a partial solution $A_s x_s = b_s$ to a linear system $Ax = b$.

1. Use A_s^{-1} to find the next row and column to append to A_s. $\rightsquigarrow O(s \log n)$
2. Compute A_{s+1}^{-1} by rank 1 updates $\rightsquigarrow O(s^2)$



- Use the vector b to compress row linear dependency information
- Improved by linear independence oracles

Lexico. search with rotations $\rightsquigarrow$ computes $\mathcal{R}^A$

[Storjohann Yang'15] Relaxed matrix inverse

Sketch of the algorithm: RowRP in $\tilde{O}(r^\omega + mn)$

1. Insted of building A_s^{-1} iteratively ($O(r^3)$), use an asymptotically fast relaxation scheme $O(r^\omega)$.
2. Requires to deal with only r columns in generic column RP.
3. Ensured by a call to [Cheung Kwok Lau'12] $+$ Toeplitz preconditionner
4. Returns the row rank profile

[Storjohann Yang'15] Relaxed matrix inverse

Sketch of the algorithm: RowRP in $O\tilde{}(r^\omega + mn)$

1. Insted of building A_s^{-1} iteratively ($O(r^3)$), use an asymptotically fast relaxation scheme $O(r^\omega)$.
2. Requires to deal with only r columns in generic column RP.
3. Ensured by a call to [Cheung Kwok Lau'12] + Toeplitz preconditionner
4. Returns the row rank profile

Problem: step 3 loses information required for the $\mathcal{R}^A$.

[Storjohann Yang'15] Relaxed matrix inverse

Sketch of the algorithm: RowRP in $\tilde{O}(r^\omega + mn)$

1. Insted of building A_s^{-1} iteratively ($O(r^3)$), use an asymptotically fast relaxation scheme $O(r^\omega)$.
2. Requires to deal with only r columns in generic column RP.
3. Ensured by a call to [Cheung Kwok Lau'12] + Toeplitz preconditionner
4. Returns the row rank profile

Problem: step 3 loses information required for the $\mathcal{R}^A$.

Solution for $\mathcal{R}^A$ in $\tilde{O}(r^\omega + mn)$

1. Compute the RowRP $\mathcal{I}$ by [Storjohann Yang'15] on A
2. Compute the ColRP $\mathcal{J}$ by [Storjohann Yang'15] on A^T

[Storjohann Yang'15] Relaxed matrix inverse

Sketch of the algorithm: RowRP in $\tilde{O}(r^\omega + mn)$

1. Insted of building A_s^{-1} iteratively ($O(r^3)$), use an asymptotically fast relaxation scheme $O(r^\omega)$.
2. Requires to deal with only r columns in generic column RP.
3. Ensured by a call to [Cheung Kwok Lau'12] $+$ Toeplitz preconditionner
4. Returns the row rank profile

Problem: step 3 loses information required for the $\mathcal{R}^A$.

Solution for $\mathcal{R}^A$ in $\tilde{O}(r^\omega + mn)$

1. Compute the RowRP $\mathcal{I}$ by [Storjohann Yang'15] on A
2. Compute the ColRP $\mathcal{J}$ by [Storjohann Yang'15] on A^T
3. Extract the $r \times r$ submatrix $A_r = A_{\mathcal{I},\mathcal{J}}$
4. Compute the LUP decomp of A_r with col. rotations

[Storjohann Yang'15] Relaxed matrix inverse

Sketch of the algorithm: RowRP in $O\tilde{}(r^\omega + mn)$

1. Insted of building A_s^{-1} iteratively ($O(r^3)$), use an asymptotically fast relaxation scheme $O(r^\omega)$.
2. Requires to deal with only r columns in generic column RP.
3. Ensured by a call to [Cheung Kwok Lau'12] $+$ Toeplitz preconditionner
4. Returns the row rank profile

Problem: step 3 loses information required for the $\mathcal{R}^A$.

Solution for $\mathcal{R}^A$ in $O\tilde{}(r^\omega + mn)$

1. Compute the RowRP $\mathcal{I}$ by [Storjohann Yang'15] on A
2. Compute the ColRP $\mathcal{J}$ by [Storjohann Yang'15] on A^T
3. Extract the $r \times r$ submatrix $A_r = A_{\mathcal{I},\mathcal{J}}$
4. Compute the LUP decomp of A_r with col. rotations
5. Recover $\mathcal{R}^A$ by inflating $\mathcal{R}^{A_r} = P$ with zeroes.

Perspective

- Application to F5 elimination (Gröbner basis) [Sun Lin Wang'14]
- Communication avoiding variants [Demmel & Al.'12]
- How to accomodate sparse elimination constraints ?
- Numerical pivoting equivalent?

Perspective

- Application to F5 elimination (Gröbner basis) [Sun Lin Wang'14]
- Communication avoiding variants [Demmel & Al.'12]
- How to accomodate sparse elimination constraints ?
- Numerical pivoting equivalent?

Thank you!

Bibliography

[Malaschonok'10]: $A = LEU$

- first instance of $\mathcal{R}^A$.
- no consideration on rank profile nor echelon form

[DSP'13]: $A = PLUQ$

Computed only via a product order pivoting,
Rank sensitive $O(r^{\omega-2}mn)$, any $m \times n$ matrix of any rank r.

[DPS'15]

- Conditions for any PLUQ alg. to reveal $\mathcal{R}^A$
- New pivoting strategies $\rightsquigarrow$ faster base case

[DPS'XX in preparation]

- $\mathcal{R}^A$ in $\tilde{O}(r^\omega + mn)$
- generalization of $\mathcal{R}^A$ to rings