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  \( \text{(LU decomposition)} \)

- 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 \( A \in \mathbb{K}^{m \times n}, r = \text{rank}(A) \).

- informally: first \( r \) linearly independent rows
- formally: lexicographically minimal list of \( r \) indices of linearly independent rows.

Example

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\( \text{Rank} = 3 \)

\( \text{RowRP} = \{1, 2, 4\} \)

\( \text{ColRP} = \{1, 2, 3\} \)

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

Generic rank profile \( \supseteq \) Generic Row RP and Generic ColRP

\[
\begin{bmatrix}
0 & 1 \\
1 & 0 \\
\end{bmatrix}
\]

\( \text{RowRP} = \text{ColRP} = \{1, 2\} \)

\[ |A_{1,1} | = 0 \]
Rank profiles

Definition (Row Rank Profile: RowRP)

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

- **Informally**: first $r$ linearly independent rows
- **Formally**: lexicographically minimal list of $r$ indices of linearly independent 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 $A \in K^{m \times n}$, $r = \text{rank}(A)$.

- **informally:** first $r$ linearly independent columns
- **formally:** lexicographic minimal list of $r$ linearly independent 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 $A \in K^{m \times n}$, $r = \text{rank}(A)$.

- **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 \text{Generic ColRP.}
Rank profiles

**Definition (Column Rank Profile: ColRP)**

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

- **Informally**: first $r$ linearly independent columns
- **Formally**: lexicographical list of $r$ linearly independent 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 \text{Generic ColRP}.

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

**Generic rank profile $\iff$ Generic Row RP and Generic ColRP**

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

RowRP = ColRP = \{1,2\}
Definition (Column Rank Profile: ColRP)

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

- informally: first $r$ linearly independent columns
- formally: lexi-co-minimal list of $r$ linearly independent 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** $\iff$ **Generic Row RP and Generic ColRP**

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

RowRP = ColRP = $\{1,2\}$
But $|A_{1,1}| = 0$
Relation to echelon forms:

- ColRP unchanged by left multiplication with an invertible matrix

\[ \text{ColRP} = \text{pivot columns in the row echelon form} \]
Triangular Matrix decompositions and rank profiles

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Exists for</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A = LU$</td>
<td>Generic rank profile</td>
<td>$Y$</td>
</tr>
</tbody>
</table>
# Triangle Matrix Decompositions and Rank Profiles

AriC Seminar. Oct 8, 2015 5 / 34

---

## Decomposition

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Exists for</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A = LU$</td>
<td>Generic rank profile</td>
<td>Y</td>
</tr>
<tr>
<td>$A = LP$</td>
<td>Generic row rank profile</td>
<td>N</td>
</tr>
<tr>
<td>$A = PL$</td>
<td>Generic col rank profile</td>
<td>N</td>
</tr>
</tbody>
</table>

---

The Rank Profile Matrix

C. Pernet (U. Grenoble Alpes)
## Triangular Matrix decompositions and rank profiles

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Exists for</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A = LU$</td>
<td>Generic rank profile</td>
<td>Y</td>
</tr>
<tr>
<td>$A = LP$</td>
<td>Generic row rank profile</td>
<td>N</td>
</tr>
<tr>
<td>$A = PL$</td>
<td>Generic col rank profile</td>
<td>N</td>
</tr>
<tr>
<td>$A = LPQ$</td>
<td>Any matrix</td>
<td>N</td>
</tr>
</tbody>
</table>
## Triangular Matrix decompositions and rank profiles

<table>
<thead>
<tr>
<th>Decomposition</th>
<th>Exists for</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A = LU$</td>
<td>Generic rank profile</td>
<td>Y</td>
</tr>
<tr>
<td>$A = LP$</td>
<td>Generic row rank profile</td>
<td>N</td>
</tr>
<tr>
<td>$A = PQ$</td>
<td>Generic col rank profile</td>
<td>N</td>
</tr>
<tr>
<td>$A = PQ$</td>
<td>Any matrix</td>
<td>N</td>
</tr>
</tbody>
</table>

$\rightarrow P, Q$ 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?
Introduction

Computing rank profiles

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

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

Let $A \in \mathbb{F}^{m \times n}$.

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 $A \in \mathbb{F}^{m \times n}$.
There exists a unique, $m \times n$, $\text{rank}(A)$-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

$$
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 \\
1 & 3 & 2 & 0 \\
2 & 5 & 4 & 7
\end{bmatrix}
\quad
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$
The rank profile Matrix

**Theorem**

Let \( A \in \mathbb{F}^{m \times n} \).

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

**Example**

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 \\
1 & 3 & 2 & 0 \\
2 & 5 & 4 & 7
\end{bmatrix}
\quad \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Theorem

Let \( A \in \mathbb{F}^{m \times n} \).

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

Example

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 \\
1 & 3 & 2 & 0 \\
2 & 5 & 4 & 7
\end{bmatrix}
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
The rank profile Matrix

**Theorem**

Let \( A \in \mathbb{F}^{m \times n} \).

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

**Example**

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

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
Properties of the rank profile matrix

Properties

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

\[
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 \\
1 & 3 & 2 & 0 \\
2 & 5 & 4 & 7 \\
\end{pmatrix}
\]

RowRP = $\{1, 3, 4\}$

\[
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

ColRP = $\{1, 2, 4\}$
Properties of the rank profile matrix

Properties

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

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 \\
1 & 3 & 2 & 0 \\
2 & 5 & 4 & 7 \\
\end{bmatrix}
\begin{align*}
\text{RowRP} &= \{1, 3\} \\
\text{ColRP} &= \{1, 2\}
\end{align*}
\]

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
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
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 $Q$
Computing the rank profile matrix

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 $R^A$?
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

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

Example

Search: “Any non zero element on the topmost row”:

\[
A = \begin{bmatrix}
2 & 0 & 3 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 4 & 0 \\
0 & 2 & 0 & 1
\end{bmatrix}
\]

\[
\Rightarrow R_A = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}, \quad \Pi_{P,Q} = \begin{bmatrix}
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}.
\]

\[
\Rightarrow \text{RowRP} = \{1, 2, 4\}.
\]
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”:

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

\[\Rightarrow \text{RowRP} = \{1,2,4\}\]
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

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]
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

\[ A = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \Rightarrow R^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \Pi_{P,Q} \]
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 \[ A = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix} \sim \mathcal{R}^A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \text{ and } \Pi_{P,Q} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \]
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 \[
A = \begin{bmatrix}
0 & 0 & 1 \\
2 & 3 & 0
\end{bmatrix}
\sim \mathcal{R}^A = \begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & 0
\end{bmatrix}
\text{ and } \Pi_{P,Q} = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0
\end{bmatrix}.
\]

\sim Pivot Swaps mix-up precedence between rows/cols.
\sim \textbf{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**

---

Cyclic rotation
Pivoting strategies revealing rank profiles

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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>$R^A$</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Pivoting strategies revealing rank profiles

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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>$\mathcal{R}^A$</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>[IMH82] [JPS13]</td>
</tr>
<tr>
<td>Col. order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $\text{RowRP} = \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix} P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
### Pivoting strategies revealing rank profiles

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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>$\mathcal{R}^A$</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>[IMH82] [JPS13] [KG85] [JPS13]</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- RowRP = $[1 \ 2 \ldots \ m] \ P \ [I_r \ 0]$
- 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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>( \mathcal{R}^A )</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>[IMH82] [JPS13]</td>
<td>[KG85] [JPS13]</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>[Sto00]</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>[Sto00]</td>
</tr>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- RowRP = \[
\begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix} P \begin{bmatrix} I_r \end{bmatrix}
\]
- ColRP = \[
\begin{bmatrix} I_r & 0 \end{bmatrix} Q \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix}^T
\]
### Pivoting strategies revealing rank profiles

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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>$\mathcal{R}_A$</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>[IMH82] [JPS13]</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td>[KG85] [JPS13]</td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>[Sto00]</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td>✓</td>
<td></td>
<td>[Sto00]</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[DPS13]</td>
</tr>
</tbody>
</table>

- $\text{RowRP} = \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix} P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- $\text{ColRP} = \begin{bmatrix} I_r & 0 \end{bmatrix} Q \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix}^T$
- $\mathcal{R}_A = P \begin{bmatrix} I_r \\ 0 \end{bmatrix} Q$
Pivoting strategies revealing rank profiles

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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>$\mathcal{R}^A$</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>[IMH82] [JPS13]</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td>[KG85] [JPS13]</td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✔</td>
<td></td>
<td></td>
<td>[Sto00]</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td>✔</td>
<td></td>
<td>[Sto00]</td>
</tr>
</tbody>
</table>

- **Product**
  - RowRP = $\begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix} P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
  - ColRP = $\begin{bmatrix} I_r & 0 \end{bmatrix} Q \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix}^T$
  - $\mathcal{R}^A = P \begin{bmatrix} I_r \\ 0 \end{bmatrix} Q$
Pivoting strategies revealing rank profiles

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

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>$\mathcal{R}^A$</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>[IMH82] [JPS13]</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td>[KG85] [JPS13]</td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[Sto00]</td>
</tr>
<tr>
<td></td>
<td>Transposition</td>
<td>Rotation</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>[DPS15]</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[DPS15]</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[Sto00]</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>Transposition</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>[DPS15]</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[DPS15]</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>[DPS15]</td>
</tr>
<tr>
<td></td>
<td>Transposition</td>
<td>Rotation</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>[DPS15]</td>
</tr>
<tr>
<td></td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[DPS13]</td>
</tr>
</tbody>
</table>

- $\text{RowRP} = \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix} P \begin{bmatrix} I_r \\ 0 \end{bmatrix}$
- $\text{ColRP} = \begin{bmatrix} I_r & 0 \end{bmatrix} Q \begin{bmatrix} 1 & 2 & \ldots & m \end{bmatrix}^T$
- $\mathcal{R}^A = P \begin{bmatrix} I_r \\ 0 \end{bmatrix} Q$
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
The slab recursive algorithm

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

1. Split $A$ Row-wise

$$
\begin{bmatrix}
A1 \\
\hline
A2
\end{bmatrix}
$$
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. $G \leftarrow A_{21} U_1^{-1} \text{(trsm)}$
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

\[ \text{TRSM: } B \leftarrow L^{-1}B \]
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: $C ← C - A \times B$
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

\[ \text{TRSM: } B \leftarrow BU^{-1} \]
The tiled recursive algorithm

Dumas, P. and Sultan 13

\[ \text{TRSM: } B \leftarrow L^{-1}B \]
The tiled recursive algorithm

Dumas, P. and Sultan 13

\[ C \leftarrow C - A \times B \]
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

fewer modular reductions than slab algorithms
rank deficiency introduces parallelism
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
  - X many permutations
  - X 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 Gflops

Implemented in FFLAS-FFPACK (kernel of LinBox).

C. Pernet (U. Grenoble Alpes)
Algorithmic instances

Iterative algorithms

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)
Algorithmic instances

Iterative algorithms

PLUQ base cases mod 131071. Rank = n/2. on an 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 Gflops
Algorithmic instances

Iterative algorithms

PLUQ base cases mod 131071. Rank = n/2. on an 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 Gflops improvement
Implemented in FFLAS-FFPACK (kernel of LinBox).

C. Pernet (U. Grenoble Alpes)
Algorithmic instances

Iterative algorithms

PLUQ base cases mod 131071. Rank = n/2. on an 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 Gflops improvement

Implemented in FFLAS-FFPACK (kernel of LinBox).

C. Pernet (U. Grenoble Alpes)

The Rank Profile Matrix

AriC Seminar. Oct 8, 2015
- > 2 Gfops improvement
- Implemented in FFLAS–FFPACK (kernel of LinBox).
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
Malaschonok LEU decomposition

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

- $E$ is an $r$-sub-permutation matrix
- Designed to avoid permutations
- $\frac{17}{2^{\omega-4}} \cdot MM(m, n)$ 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

\[
E = \mathcal{R}^A
\]

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

\[ E = \mathcal{R}^A \]

\[
A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T P \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} QQ^T \begin{bmatrix} U & V \\ I_{n-r} \end{bmatrix} Q
\]
Malaschonok LEU decomposition

[Malaschonok’10]: \( A = L \cdot E \cdot U \)

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

Fact

\[ E = \mathcal{R}^A \]

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

\( E = R^A \)

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

With appropriate pivoting: \( \Pi_{P,Q} = R(A) \)
LUP and PLU decompositions

**LUP**

If \( A \) has generic RowRP

- \( LUP(A) \) with Lex order and col. rot.: 

\[
\begin{align*}
& \rightsquigarrow \begin{bmatrix} I_r & 0 \end{bmatrix} P = \mathcal{R}^A \\
& \text{In particular, if } A \text{ has full row rank and } m = n:
& \rightsquigarrow P = \mathcal{R}^A
\end{align*}
\]
LUP and PLU decompositions

### LUP

If $A$ has generic RowRP

- $LUP(A)$ with Lex order and col. rot.: $\sim [I_r \ 0] P = \mathcal{R}^A$

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

### PLU

If $A$ has generic ColRP

- $PLU(A)$ with RevLex order and row rot. $\sim P [I_r \ 0] = \mathcal{R}^A$

In particular, if $A$ has full column rank and $m = n$: $\sim P = \mathcal{R}^A$
Relations to other decompositions

Bruhat decomposition

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

\[ \mathcal{R}_A = P \quad Q \]

for

\[ C = P \text{L} \quad U \quad Q \]

\[ C = P \text{L} P_s \]

\[ Q_s U Q = E \]
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 $r \ll m, n$, $O(mnr^{-2})$ can be too expensive. (Compressed sensing applications)

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

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

[Storjohann Yang’14:] Rank profile in $O^\sim(r^3 + mn)$ probabilistic.

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

[Storjohann Yang’14: ] Rank profile in $O^\sim(r^3 + mn)$ probabilistic.

[Storjohann Yang’15: ] Rank profile in $O^\sim(r^\omega + mn)$ probabilistic.

Can the rank profile matrix be computed in such complexities?
Sketch of the $O(\tilde{r}^3 + mn)$ 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 $Ax = b$.
Sketch of the $O(\tilde{r}^3 + mn)$ 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 $A_s$. $\leadsto O(sn)$
Sketch of the $O( r^3 + mn )$ 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 $A_s$. $\sim O(sn)$
Sketch of the $O(\tilde{r}^3 + mn)$ 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 $A_s$. $\sim O(sn)$
Sketch of the $O(r^3 + mn)$ 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 $A_s$. $\sim O(sn)$

\[ \begin{bmatrix} 0 & 0 \\ \cdot & \cdot \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]
Sketch of the $O(\tilde{r}^3 + mn)$ 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 $A_s$. $\sim O(sn)$
2. Compute $A_{s+1}^{-1}$ by rank 1 updates $\sim O(s^2)$
Sketch of the $O(\tilde{r}^3 + mn)$ 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 $A_s$. $\sim O(sn)$
2. Compute $A_{s+1}^{-1}$ by rank 1 updates $\sim O(s^2)$

- Use the vector $b$ to compress row linear dependency information
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 $A_s$. $\sim O(s \log n)$
2. Compute $A_{s+1}^{-1}$ by rank 1 updates $\sim O(s^2)$

- Use the vector $b$ to compress row linear dependency information
- Improved by linear independence oracles
Sketch of the $O(r^3 + mn)$ 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 $A_s$.  \( \leadsto O(s \log n) \)
2. Compute $A_{s+1}^{-1}$ by rank 1 updates \( \leadsto O(s^2) \)

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

Lexico. search with rotations $\leadsto$ computes $\mathcal{R}^A$
[Storjohann Yang’15] Relaxed matrix inverse

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

1. Instead 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 $O^\sim(r^\omega + mn)$

1. Instead 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 $R^A$. 
[Storjohann Yang’15] Relaxed matrix inverse

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

1. Instead 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 preconditioner
4. Returns the row rank profile

Problem: step 3 loses information required for the $R^A$.

Solution for $R^A$ in $O^\sim(r^\omega + mn)$

1. Compute the RowRP $I$ by [Storjohann Yang’15] on $A$
2. Compute the ColRP $J$ by [Storjohann Yang’15] on $A^T$
The small rank case

[Storjohann Yang’15] Relaxed matrix inverse

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

1. Instead 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 preconditioner
4. Returns the row rank profile

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

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

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
The small rank case

[Storjohann Yang’15] Relaxed matrix inverse

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

1. Instead 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.
4. Returns the row rank profile.

Problem: step 3 loses information required for the $R^A$.

Solution for $R^A$ in $O^\sim(r^\omega + mn)$

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 $R^A$ by inflating $R^{A_r} = P$ with zeroes.
Perspective

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

Thank you!
[Malaschonok’10]: $A = LEU$

- first instance of $R^A$.
- 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 $R^A$
- New pivoting strategies $\Rightarrow$ faster base case

[DPS’XX in preparation]

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