Simultaneous computation of the row and column rank profiles

J-G. DUMAS, C. PERNET and Z. SULTAN

University J. Fourier (Grenoble), INRIA, CNRS, LIG, LJK

July 17, 2013

ISSAC 2013, Boston.

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)

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)

Linear dependencies and row/column rank profiles

Definition (Row Rank Profile: RowRP)

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

informally: *first* r linearly independent rows

formally: lexico-minimal sub-sequence of $(1, \ldots, m)$ of r indices of linearly independant rows.

Example

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

Linear dependencies and row/column rank profiles

Definition (Row Rank Profile: RowRP)

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

informally: *first* r linearly independent rows

formally: lexico-minimal sub-sequence of $(1, \ldots, m)$ of r indices of linearly independant rows.

Example

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

Rank = 3

RowRP = $\{$*1,2,4*$\}$

Linear dependencies and row/column rank profiles

Definition (Row Rank Profile: RowRP)

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

informally: *first* r linearly independent rows

formally: lexico-minimal sub-sequence of $(1,\ldots,m)$ of r indices of linearly independant rows.

Example

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

Rank = 3

RowRP = {1,2,4}

ColRP = {1,2,3}

Linear dependencies and row/column rank profiles

Definition (Row Rank Profile: RowRP)

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

informally: *first* r linearly independent rows

formally: lexico-minimal sub-sequence of $(1,\ldots,m)$ of r indices of linearly independant rows.

Example

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

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

Generic Row Rank Profile : if it equals $\{1,\ldots,r\}$.

Linear dependencies and row/column rank profiles

Definition (Row Rank Profile: RowRP)

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

informally: *first* r linearly independent rows

formally: lexico-minimal sub-sequence of $(1, \ldots, m)$ of r indices of linearly independant rows.

Example

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

Rank = 3

RowRP = $\{$*1,2,4*$\}$

ColRP = $\{$*1,2,3*$\}$ $\rightarrow$ *Generic ColRP.*

Generic Row Rank Profile : if it equals $\{1, \ldots, r\}$.

Generic rank profile

Definition (Generic Rank Profile)

First r leading principal minors $\neq 0$

Example

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

Rank = 4

RowRP = ColRP = $\{1,2,3,4\}$

Generic rank profile

Definition (Generic Rank Profile)

First r leading principal minors $\neq 0$

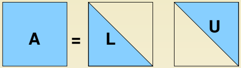
Example

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

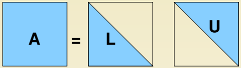
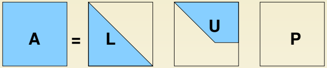
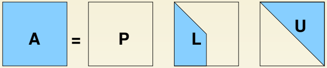
Rank = 4
RowRP = ColRP = $\{$*1,2,3,4*$\}$
But $|A_{1..2,1..2}| = 0$

$\rightarrow$ **Generic row and column rank profiles** $\not\Rightarrow$ **Generic rank profile**

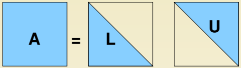
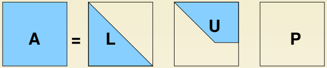
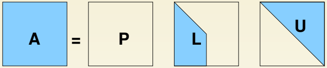
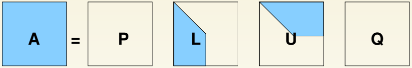
Triangular Matrix decompositions

Decomposition	Exists for	Unique
	Generic rank profile	Y

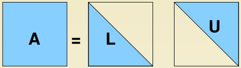
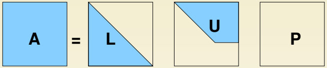
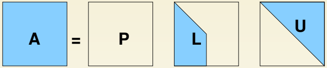
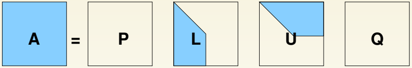
Triangular Matrix decompositions

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

Triangular Matrix decompositions

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

Triangular Matrix decompositions

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

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

PLUQ decomposition

PLUQ decomposition is not unique:
$\rightarrow$ the **pivoting strategy** impacts P, Q.

Numerical linear algebra: *full pivoting*

- choosing the largest pivot (numerical stability)
- increasing data locality
- reducing fill-in (sparse elimination)

Exact linear algebra

- No stability issue: a pivot just has to be invertible
- Some pivoting strategies can reveal the row or the col rank profile

Rank profile revealing PLUQ decomp.: state of the art

Row Rank profile

Pivoting strategy: **row-major** search for a pivot

- equiv. to LSP, LQUP decompositions [**?**]
- equiv. to CUP decomposition [**?**]
- $\mathcal{O}\left(mnr^{\omega-2}\right)$, in-place

Rank profile revealing PLUQ decomp.: state of the art

Row Rank profile

Pivoting strategy: **row-major** search for a pivot

- equiv. to LSP, LQUP decompositions [**?**]
- equiv. to CUP decomposition [**?**]
- $\mathcal{O}\left(mnr^{\omega-2}\right)$, in-place

Column rank profile

Pivoting strategy: **column-major** search for a pivot

- Searching for a pivot in a column-major fashion
- equvi. to PLE decomposition [Jeannerod & Al. 13]
- $\mathcal{O}\left(mnr^{\omega-2}\right)$, in-place

Contribution

A new pivoting strategy for PLUQ

- revealing Row **and** Column rank profiles **at the same time**
- also revealing the RowRP and ColRP of **all** n^2 leading sub-matrices.

Contribution

A new pivoting strategy for PLUQ

- revealing Row **and** Column rank profiles **at the same time**
- also revealing the RowRP and ColRP of **all** n^2 leading sub-matrices.

Algorithms

- Two rank sensitive algorithms:
  - Quad-recursive in $O(mnr^{\omega-2})$
  - Iterative in $O(mnr)$ (to be used as a base case)
- Time, memory and nb of mod reductions at least as good as state of the art CUP/PLE decompositions.
- Fast implementation

Outline

Outline

A new pivoting strategy

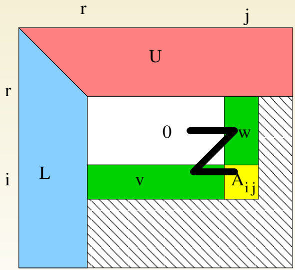
Key features:

	CUP, LSP, ...
Search for pivot:	Row after row
Pivot Permutation:	swap

A new pivoting strategy

Key features:

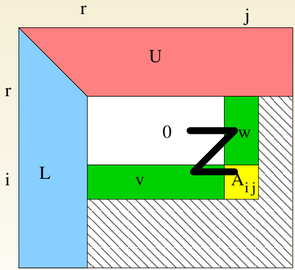
	CUP, LSP, ...	New strategy
Search for pivot:	Row after row	increasing leading sub-matrix
Pivot Permutation:	swap	



A new pivoting strategy

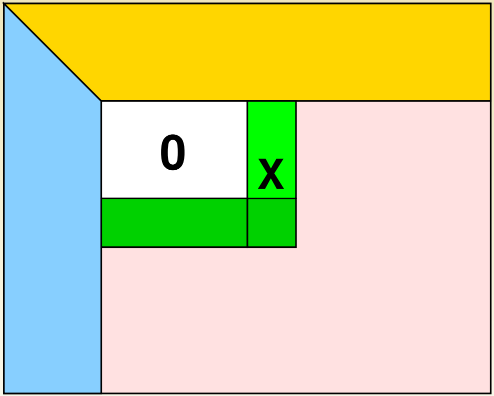
Key features:

	CUP, LSP, ...	New strategy
Search for pivot:	Row after row	increasing leading sub-matrix
Pivot Permutation:	swap	order 1 rotation

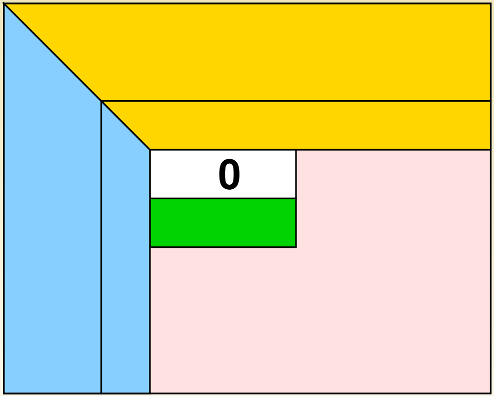


Outline

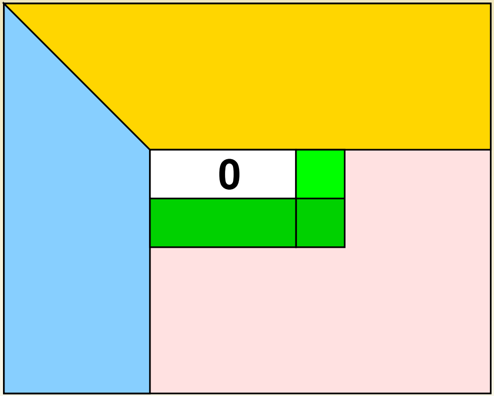
PLUQ iterative version



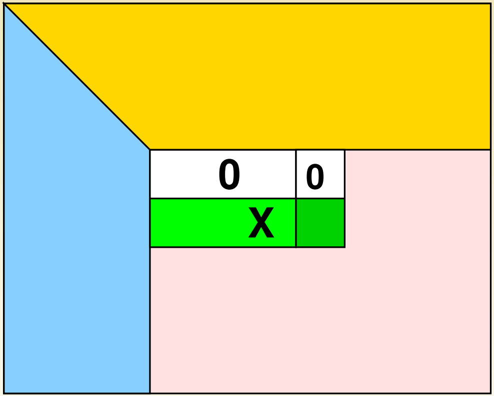
PLUQ iterative version



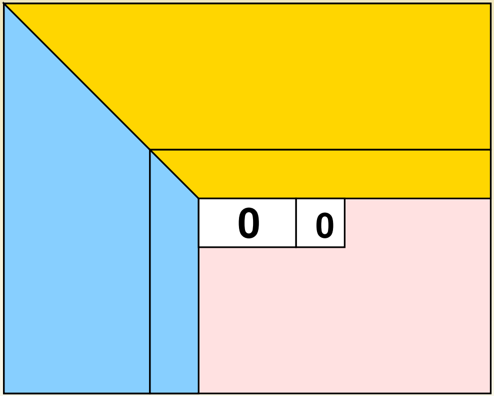
PLUQ iterative version



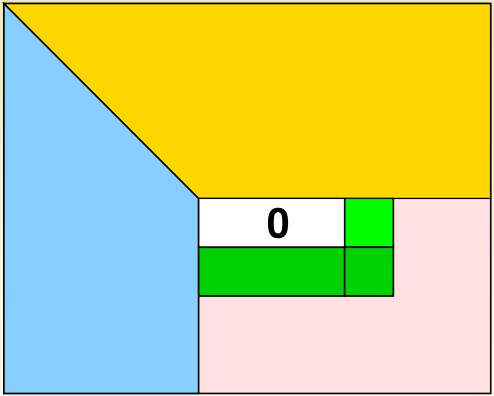
PLUQ iterative version



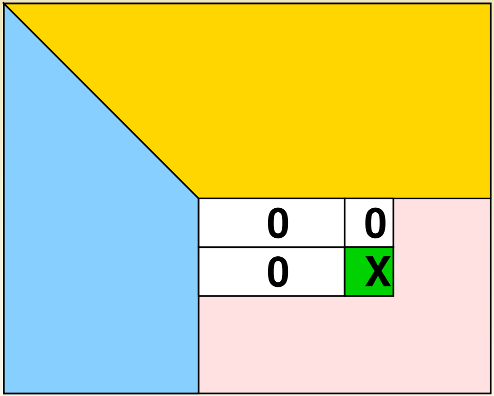
PLUQ iterative version



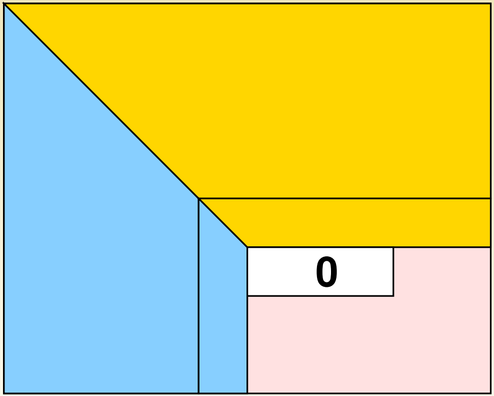
PLUQ iterative version



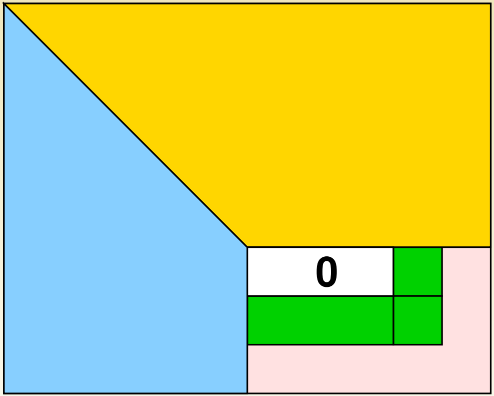
PLUQ iterative version



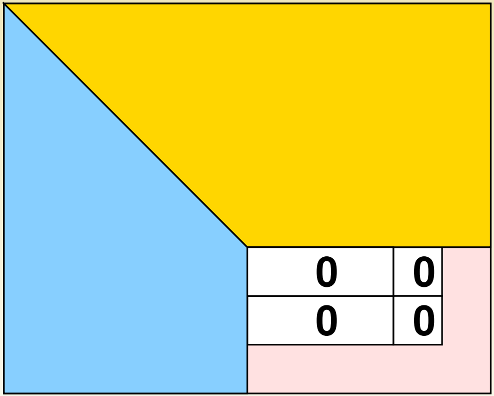
PLUQ iterative version



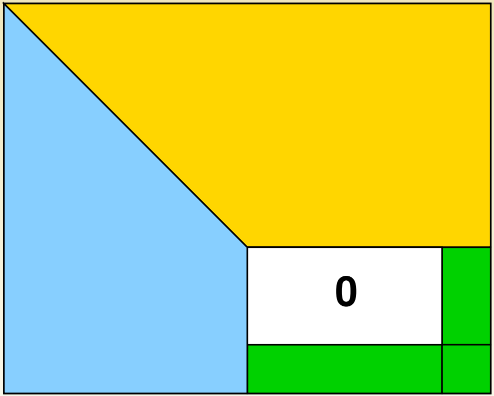
PLUQ iterative version



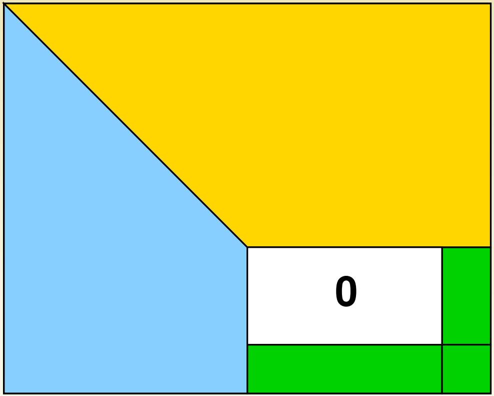
PLUQ iterative version



PLUQ iterative version

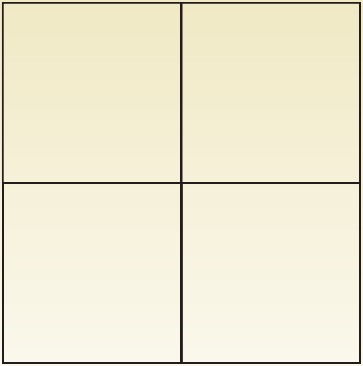


PLUQ iterative version



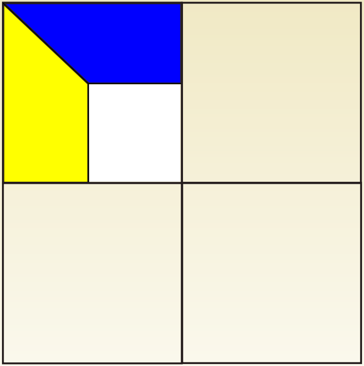
- Complexity $\mathcal{O}(mnr)$
- Further optimizations:
  - Left-looking variant
  - All permutations delayed to the end
- To be used as a base case

Quad-recursive PLUQ algorithm



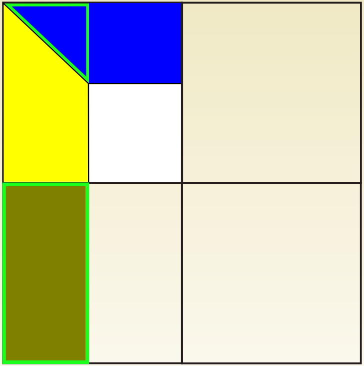
2×2 block splitting

Quad-recursive PLUQ algorithm



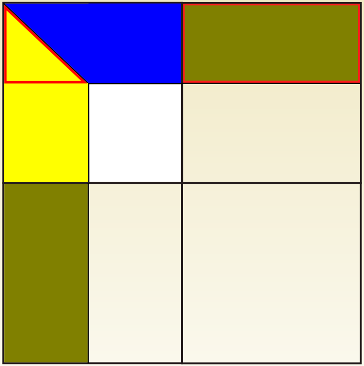
Recursive call

Quad-recursive PLUQ algorithm



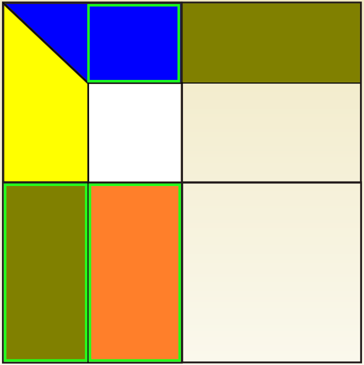
TRSM: $B \leftarrow BU^{-1}$

Quad-recursive PLUQ algorithm



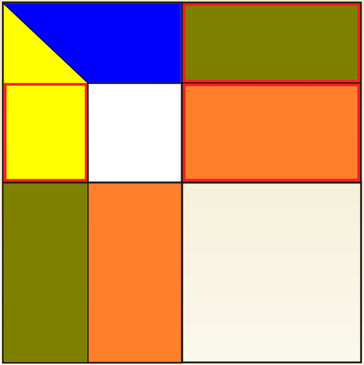
TRSM: $B \leftarrow L^{-1}B$

Quad-recursive PLUQ algorithm



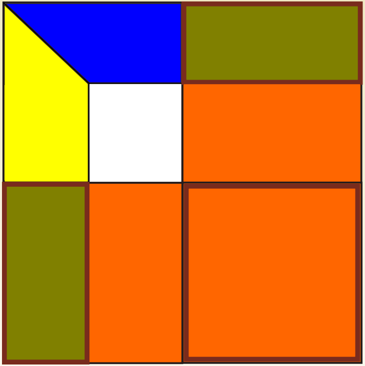
MatMul: $C \leftarrow C - A \times B$

Quad-recursive PLUQ algorithm



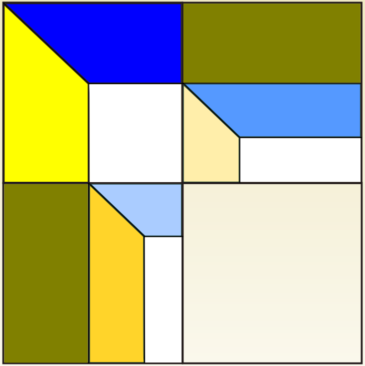
MatMul: $C \leftarrow C - A \times B$

Quad-recursive PLUQ algorithm



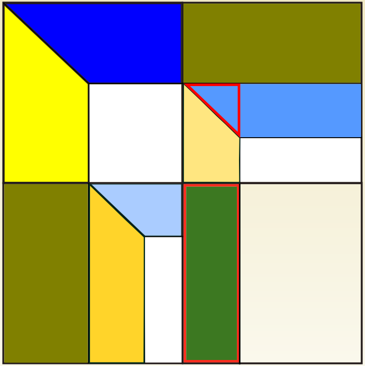
MatMul: $C \leftarrow C - A \times B$

Quad-recursive PLUQ algorithm



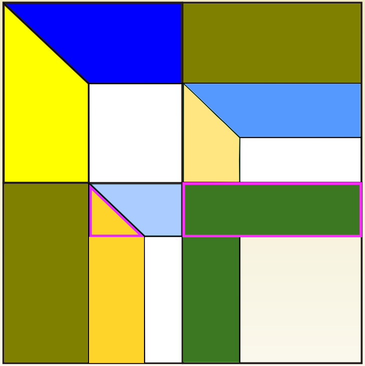
2 independent recursive calls

Quad-recursive PLUQ algorithm



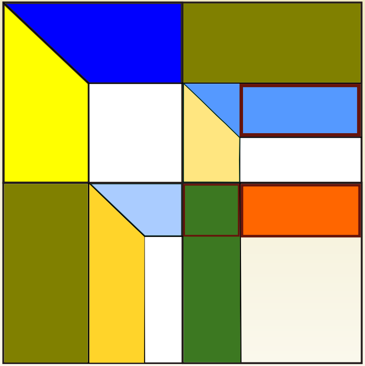
TRSM: $B \leftarrow BU^{-1}$

Quad-recursive PLUQ algorithm



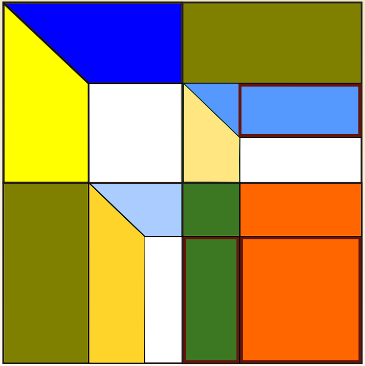
TRSM: $B \leftarrow L^{-1}B$

Quad-recursive PLUQ algorithm



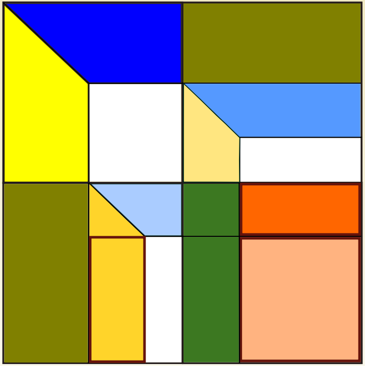
MatMul: $C \leftarrow C - A \times B$

Quad-recursive PLUQ algorithm



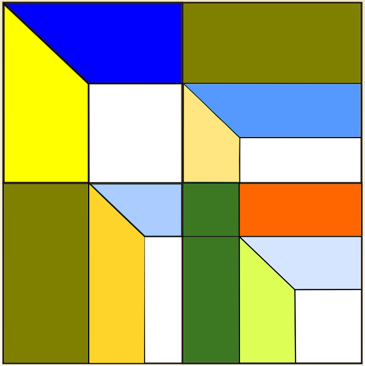
MatMul: $C \leftarrow C - A \times B$

Quad-recursive PLUQ algorithm



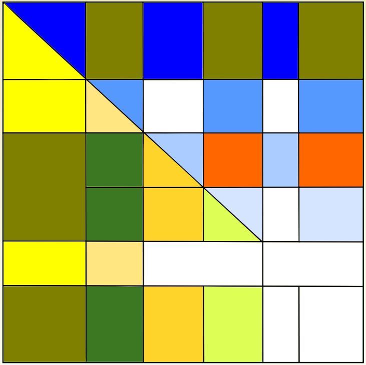
MatMul: $C \leftarrow C - A \times B$

Quad-recursive PLUQ algorithm



Recursive call

Quad-recursive PLUQ algorithm

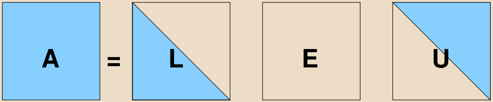


Puzzle game (block permutations)

Outline

LEU decomposition

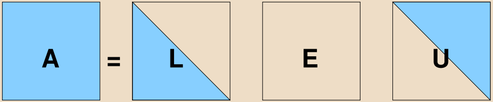
LEU decomposition [?]: for any matrix A



E is a permutation matrix with $n - r$ rows zeroed out.

LEU decomposition

LEU decomposition [?]: for any matrix A



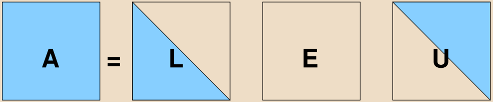
E is a permutation matrix with $n-r$ rows zeroed out.

From PLUQ to LEU

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

LEU decomposition

LEU decomposition [?]: for any matrix A



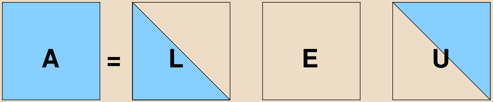
E is a permutation matrix with $n - r$ rows zeroed out.

From PLUQ to LEU

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

LEU decomposition

LEU decomposition [?]: for any matrix A



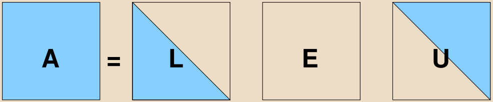
E is a permutation matrix with $n - r$ rows zeroed out.

From PLUQ to LEU

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

LEU decomposition

LEU decomposition [?]: for any matrix A



E is a permutation matrix with $n - r$ rows zeroed out.

From PLUQ to LEU

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

Theorem

$\exists$ *pivoting strategy s.t.* $\overline{L}$ *is lower triangular and* $\overline{U}$ *is upper triangular.*

Computing Rank Profiles from

Lemma

1. *RowRP and ColRP of A and E are identical*
2. *RowRP and ColRP of any leading submatrix of A and E are identical*

Example

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 8 \\ 1 & 2 & 3 & 4 \\ 3 & 5 & 9 & 12 \end{bmatrix} \longrightarrow E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Computing Rank Profiles from

Lemma

1. *RowRP and ColRP of A and E are identical*
2. *RowRP and ColRP of any leading submatrix of A and E are identical*

Example

$$A = \begin{array}{|cccc|} 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 8 \\ 1 & 2 & 3 & 4 \\ 3 & 5 & 9 & 12 \end{array} \longrightarrow E = \begin{array}{|cccc|} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}$$

RowRP = $\{1\}$
ColRP = $\{1\}$

Computing Rank Profiles from

Lemma

1. *RowRP and ColRP of A and E are identical*
2. *RowRP and ColRP of any leading submatrix of A and E are identical*

Example

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 8 \\ 1 & 2 & 3 & 4 \\ 3 & 5 & 9 & 12 \end{bmatrix} \longrightarrow E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

RowRP = {*1,2*}
ColRP = {*1,3*}

Computing Rank Profiles from

Lemma

1. *RowRP and ColRP of A and E are identical*
2. *RowRP and ColRP of any leading submatrix of A and E are identical*

Example

$$A = \left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 8 \\ 1 & 2 & 3 & 4 \\ 3 & 5 & 9 & 12 \end{array}\right] \longrightarrow E = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

RowRP = $\{1,4\}$
ColRP = $\{1,2\}$

Computing Rank Profiles from

Lemma

1. *RowRP and ColRP of A and E are identical*
2. *RowRP and ColRP of any leading submatrix of A and E are identical*

Example

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 8 \\ 1 & 2 & 3 & 4 \\ 3 & 5 & 9 & 12 \end{bmatrix} \longrightarrow E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

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

Algorithm: Row and Col Rank Profiles from (P,Q)

RowRP(A) = sort $(\{i \in \{1 \ldots m\} | P_{i,j} = 1$ for $j \in \{1 \ldots r\}\})$
ColRP(A) = sort $(\{j \in \{1 \ldots n\} | Q_{i,j} = 1$ for $i \in \{1 \ldots r\}\})$

Computing Rank Profiles from

Lemma

1. *RowRP and ColRP of A and E are identical*
2. *RowRP and ColRP of any leading submatrix of A and E are identical*

Example

A

1	2	3	4
2	4	5	8
1	2	3	4
3	5	9	12

→

E

1	0	0	0
0	0	1	0
0	0	0	0
0	1	0	0

RowRP = {*1,4*}
ColRP = {*1,2*}

Algorithm: Row and Col Rank Profiles from (P,Q)

RowRP$(A_{1\ldots c,1\ldots d})$ = sort $(\{i \in \{1 \ldots c\} | P_{i,j} = 1$ for $j \in \{1 \ldots r\}\})$
ColRP$(A_{1\ldots c,1\ldots d})$ = sort $(\{j \in \{1 \ldots d\} | Q_{i,j} = 1$ for $i \in \{1 \ldots r\}\})$

Outline

Implementation

In `fflas-ffpack` (`LinBox` kernel for dense linalg mod p)

`LinBox`: exact linear algebra

- LGPL, `http://linalg.org/`
- Floating point storage of word size finite field elements.
- Matrix Multiplication: numerical BLAS & Strassen-Winograd

Implementation

In `fflas-ffpack` (`LinBox` kernel for dense linalg mod p)

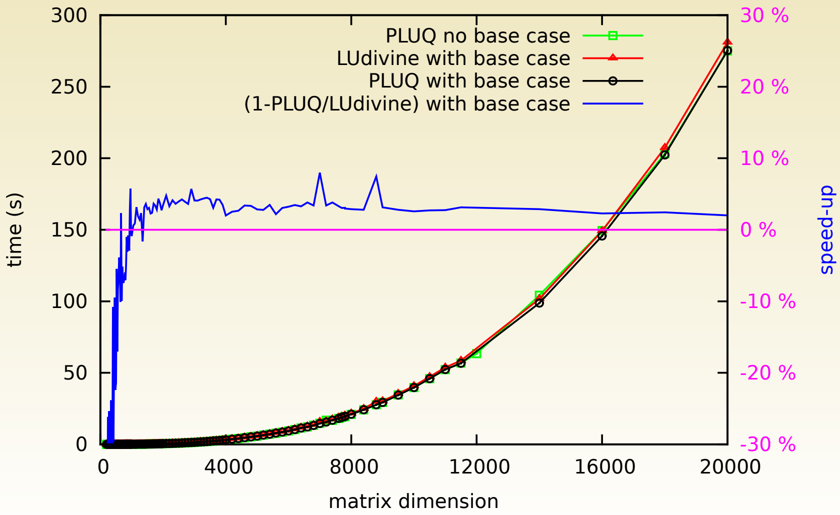
`LinBox`: exact linear algebra

- LGPL, `http://linalg.org/`
- Floating point storage of word size finite field elements.
- Matrix Multiplication: numerical BLAS & Strassen-Winograd

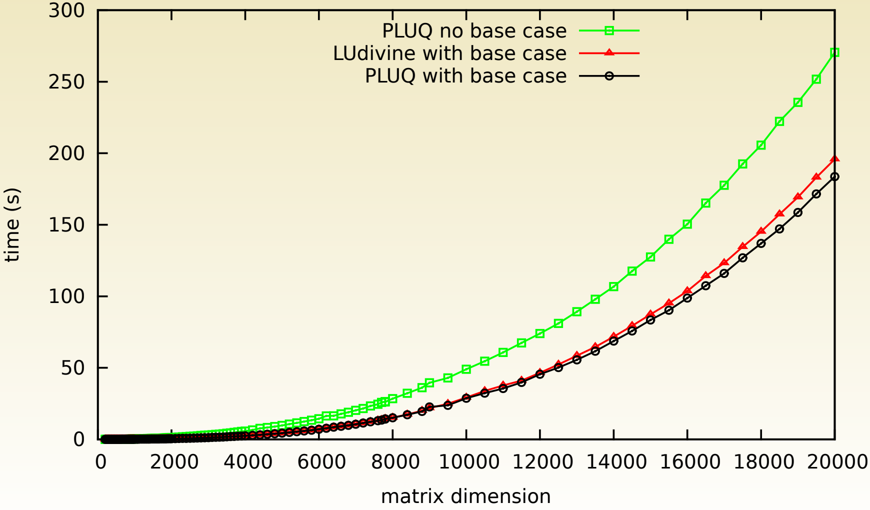
Experiments

- compared to state of the art implementation of CUP (`LUdivine`)
- Experimentally tuned base-case threshold.
- Intel SandyBridge E5-4620 2.2Ghz, 32 cores, L3(16MB).
- gcc-4.7.3, OpenBLAS-93dd133
- Two classes of square matrices over $\mathbb{Z}/1009\mathbb{Z}$
  - generic rank profiles,
  - arbitrary rank profiles: from LEU with rank $= n/2$, and pivots of E uniformly distributed

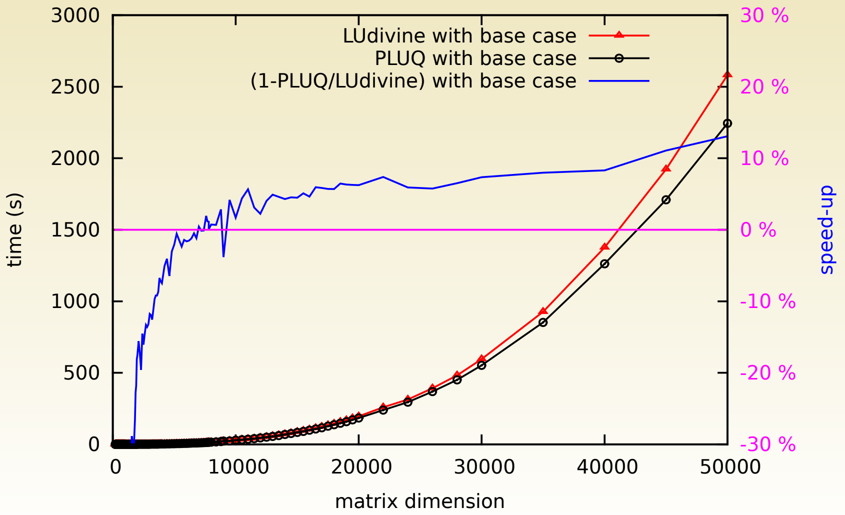
Generic rank profile matrices



Arbitrary rank profile matrices



Arbitrary rank profile matrices (larger dimension)



Outline

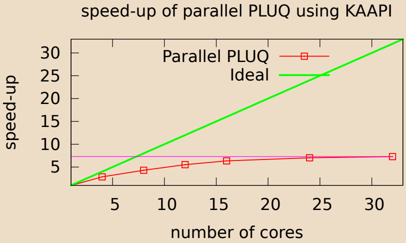
Conclusion

- A new pivoting strategy
- Two algorithms
  - $\mathcal{O}\left(mnr^{\omega-2}\right)$ quad-recursive algorithm
  - $\mathcal{O}\left(mnr\right)$ base case
- Rank profile recovery:
  - all rank profiles in just one elimination,
  - first reduction of **computing RowRP and ColRP of all leading submatrices** to **matrix multiplication**.
- Faster in practice than previous eliminations:
  - data locality,
  - fewer modular reductions.

Perspective

Parallelization

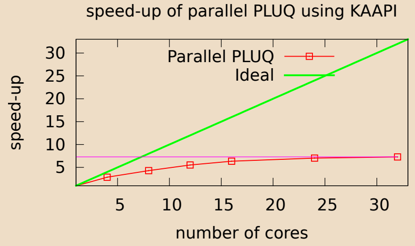
- Rank deficiency becomes an advantage:
- First toy implementation with tasks:



Perspective

Parallelization

- Rank deficiency becomes an advantage:
- First toy implementation with tasks:



Next step

- Tiled iterative version
  - less synchronizations
  - Challenge: n^4 tasks, most of which are empty
- Data-flow task dependencies in the recursive algorithm

Thank you

References