Vectorization of GE

(kij)-form (standard form):

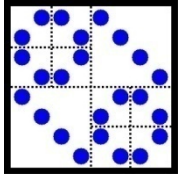
For $k = 1 : n-1$
  For $i = k+1 : n$
    $l_{i,k} = a_{i,k} / a_{k,k}$;
  end
  For $i = k+1 : n$
    For $j = k+1 : n$
      $a_{i,j} = a_{i,j} - l_{i,k}\, a_{k,j}$;
    end
  end
end

} Vector operation $\alpha \cdot \vec{x}$

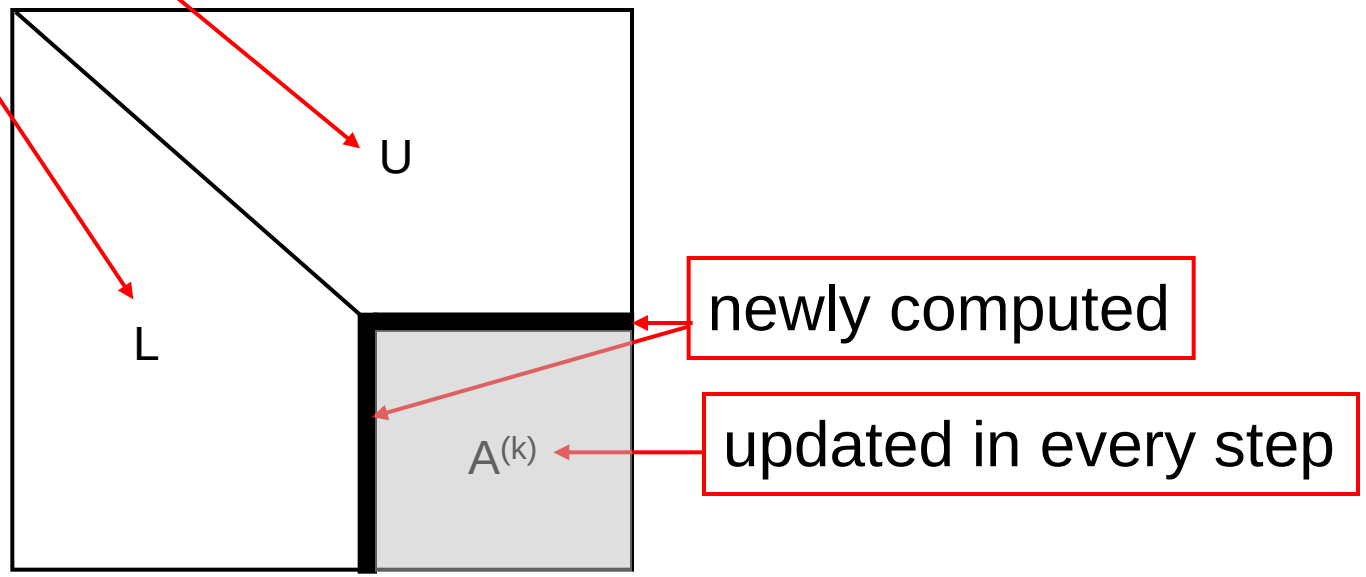
} SAXPY in rows a_i and a_k

} No GAXPY

U computed rowwise, L columnwise.



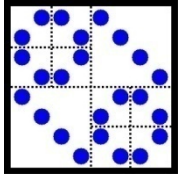
already computed, remains unchanged, not used anymore



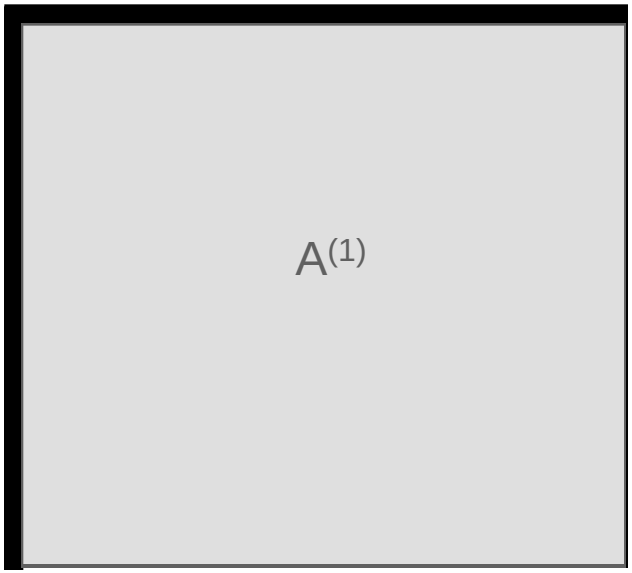
newly computed

updated in every step

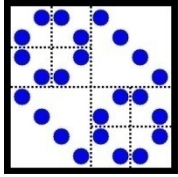
Standard (kij) form is also called “rightlooking GE”.



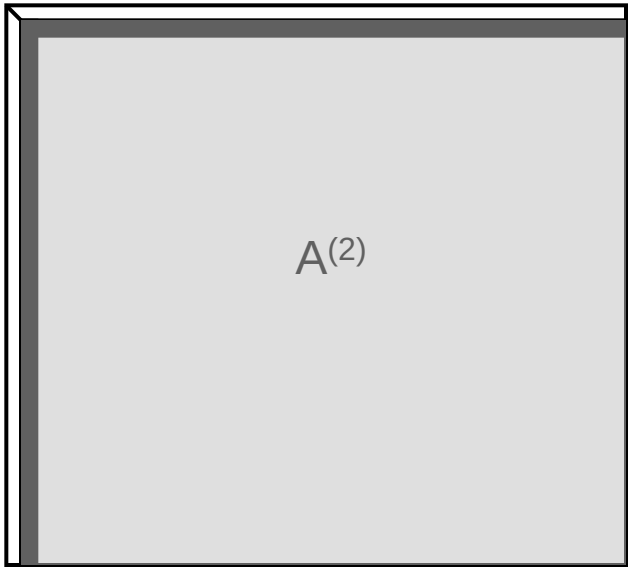
First Elimination step:



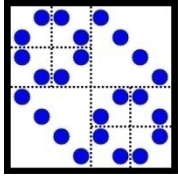
Compute first column of L
Update $A^{(1)}$



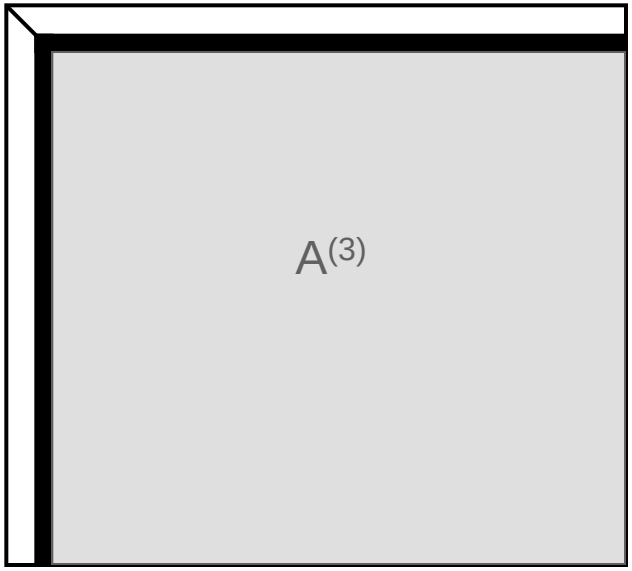
Second step:



Compute second column of L
Update $A^{(2)}$

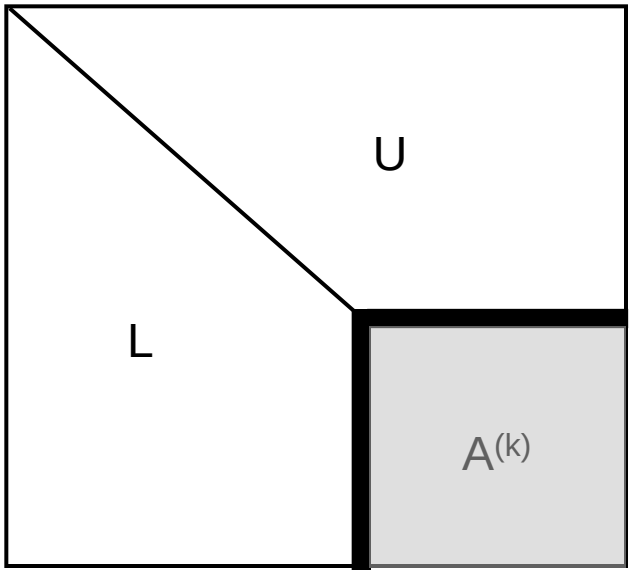


Second step:

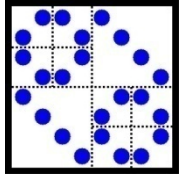


Compute third column of L
Update $A^{(3)}$

k-1st step:



Compute k-th column of L
Update $A^{(k)}$



Rules for different i,j,k forms:

In the following we again interchange the kij loops.

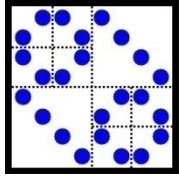
Necessary conditions: $1 \le k < i \le n$

$1 \le k < j \le n$

Furthermore:
Innermost index i,j, or k determines whether the computation is done row, column, or block-wise.

Outermost index shows how the final parts are derived.

Weights l_{jk} have to be computed before they are used to eliminate related entries.



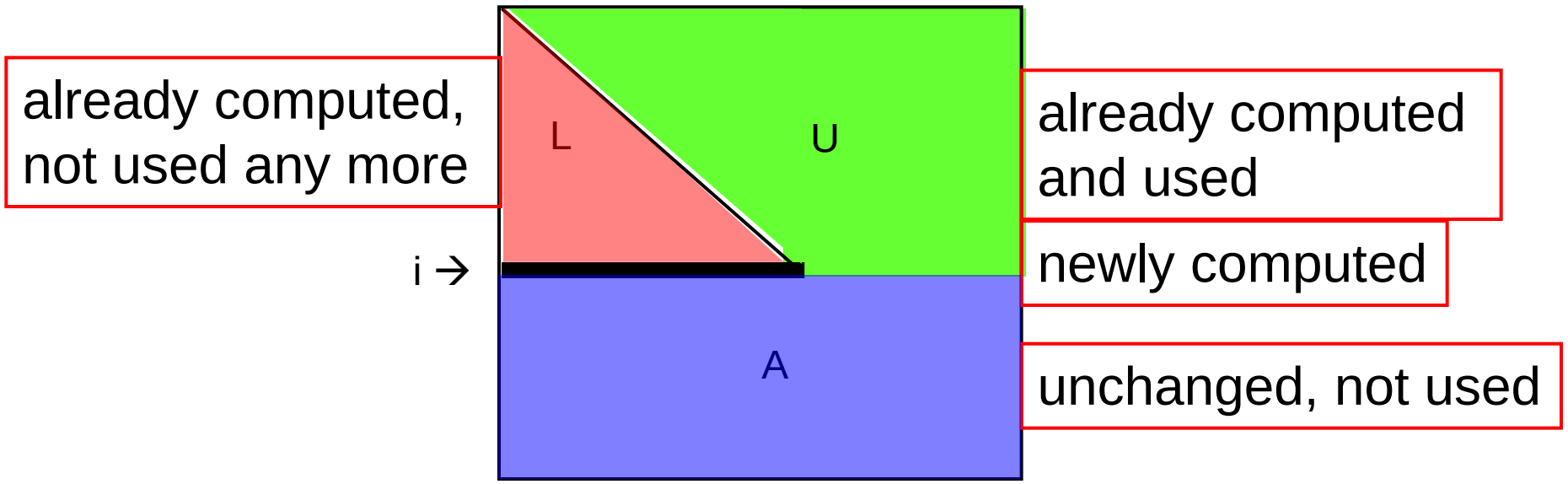
$$1 \le k < i \le n$$

$$1 \le k < j \le n$$

(ikj)-form:

For i = 2 : n
  For k = 1 : i-1
    $l_{i,k} = a_{i,k} / a_{k,k}$;
    For j = k+1 : n
      $a_{i,j} = a_{i,j} - l_{i,k}\, a_{k,j}$;
    end
  end
end

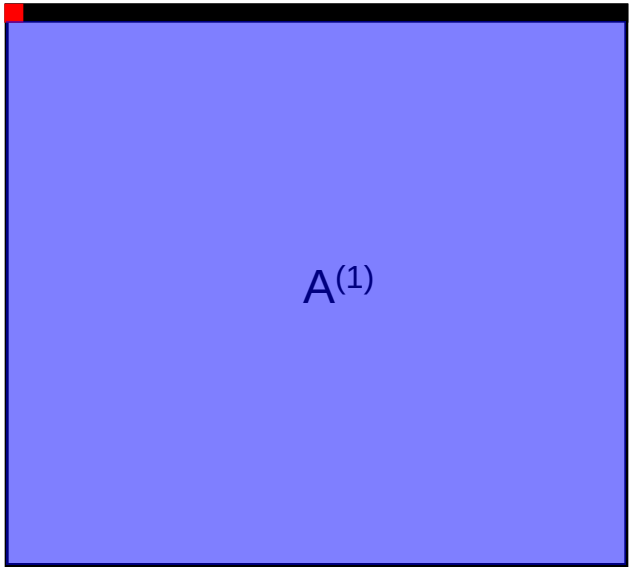
} GAXPY in $a_{.i}$



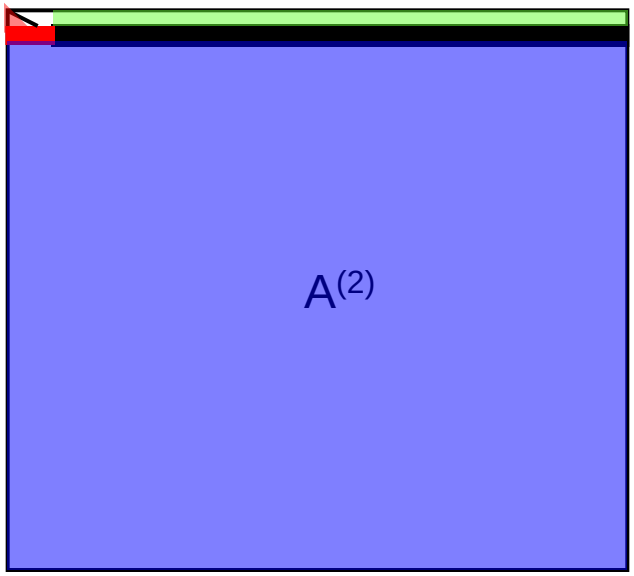
L and U computed rowwise.

Compute $l_{i,1}$, then SAXPY for 1st and i-th row;
then $l_{i,2}$ and so on …

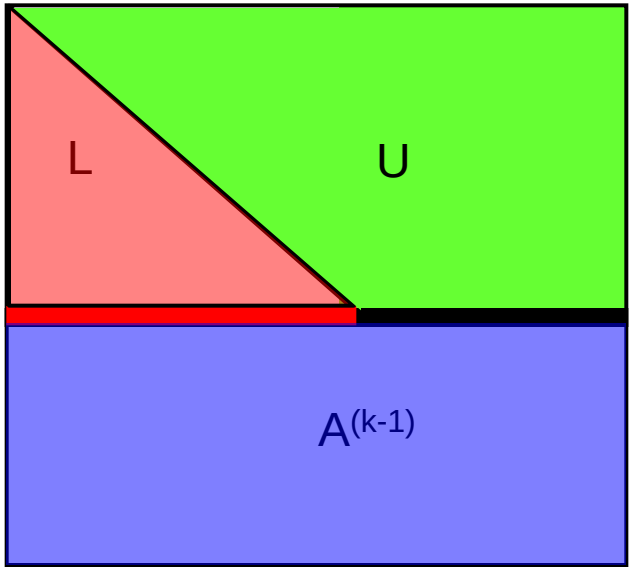
First step



Second step



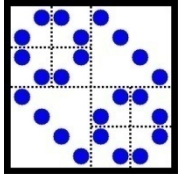
k-1-st step



(ijk)-form:

$1 \le k < i \le n$

$1 \le k < j \le n$



For i = 2 : n

  For j = 2 : i

    $l_{i,j-1} = a_{i,j-1} / a_{j-1,j-1}$; ← new row

    For k = 1 : j-1

      $a_{i,j} = a_{i,j} - l_{i,k}\, a_{k,j}$; } Dot product — left part

    end

  end

  For j = i+1 : n

    For k = 1 : i-1

      $a_{i,j} = a_{i,j} - l_{i,k}\, a_{k,j}$; } Dot product — right part

    end

  end

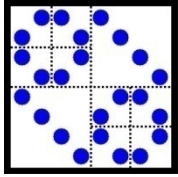
end

Compute $l_{i,1}$ and update $a_{i,2}$; then compute $l_{i,2}$ and update $a_{i,2}$ and $a_{i,3}$, ….

Accumulating $a_{i,j}$

$1 \le k < i \le n$

$1 \le k < j \le n$



(jki)-form:

For j = 2 : n

  For k = j : n

    $l_{k,j-1} = a_{k,j-1} / a_{j-1,j-1}$;

  end

  For k = 1 : j-1

    For i = k+1 : n

      $a_{i,j} = a_{i,j} - l_{i,k}\, a_{k,j}$;

    end

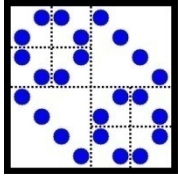
  end

end

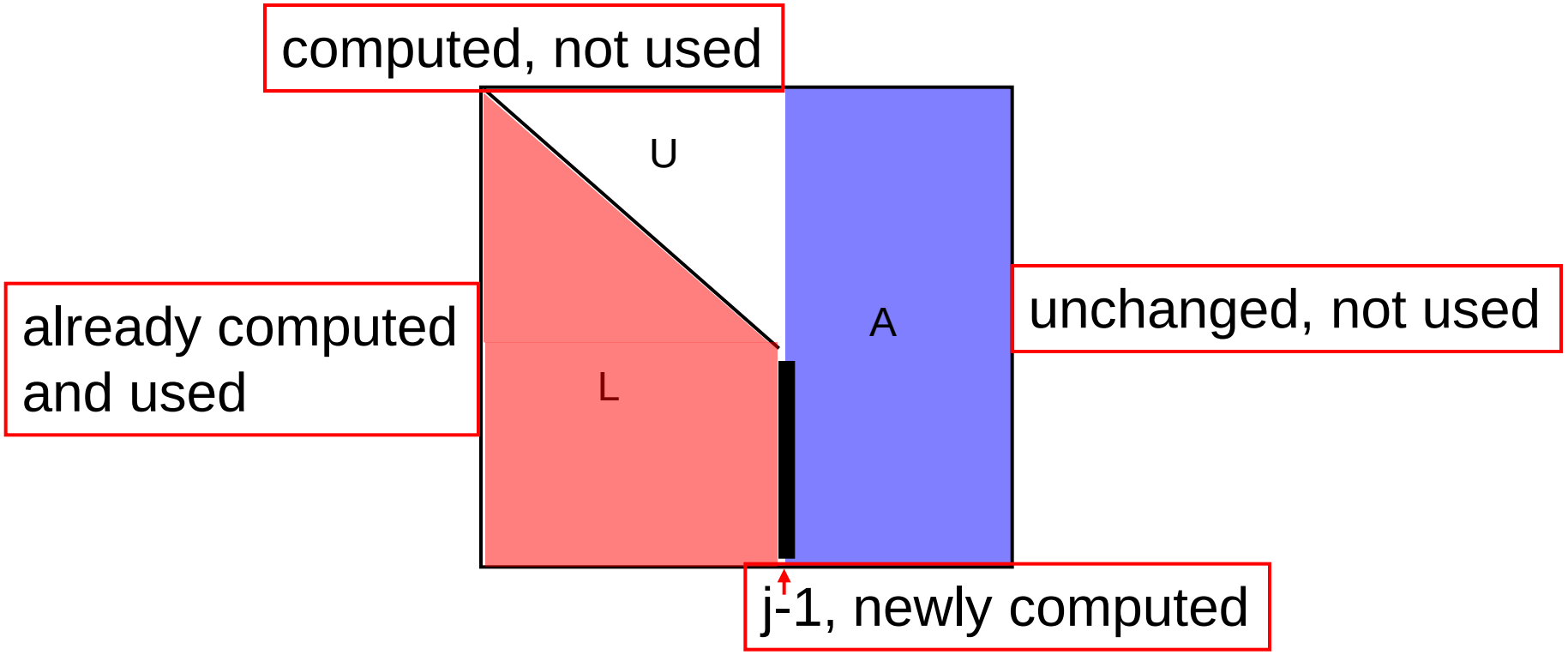
$\alpha \cdot \vec{x}$

new column of L

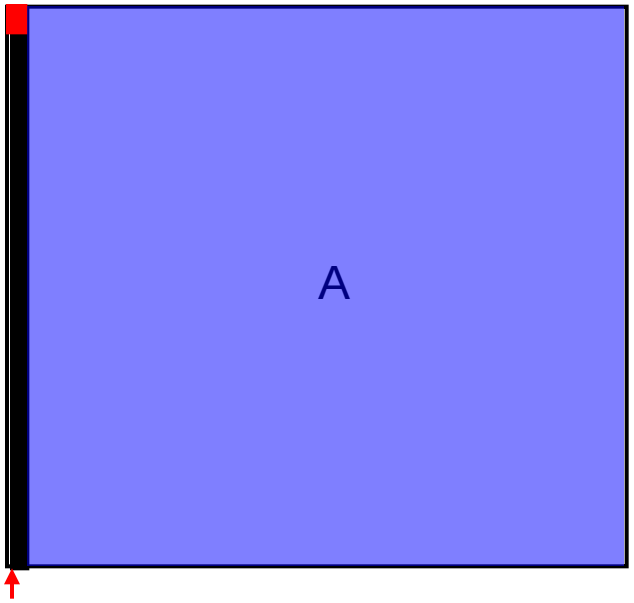
GAXPY in $a_{.i}$



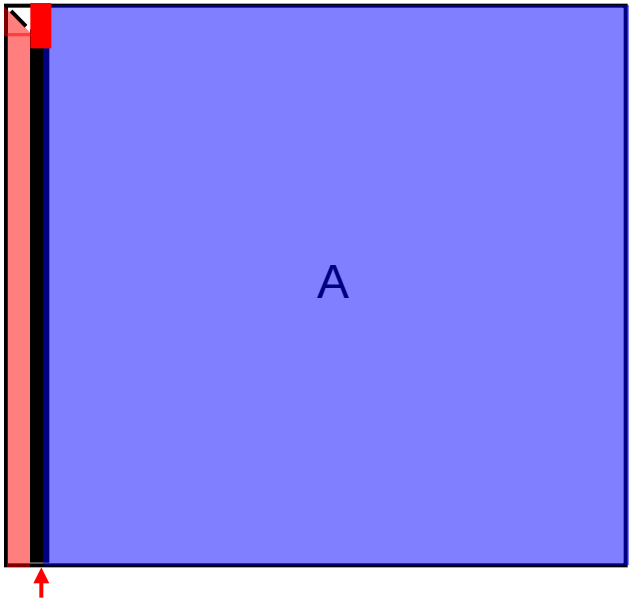
Left looking GE



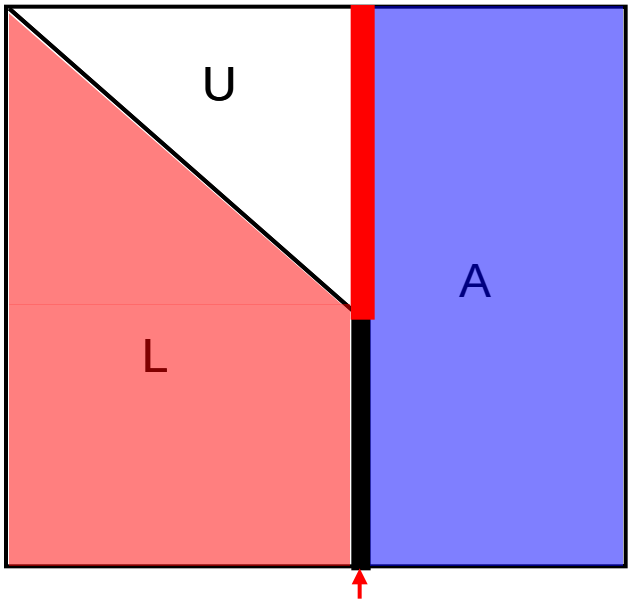
First step



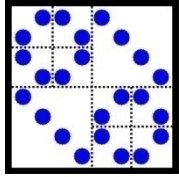
Second step



k-1-st step



Overview



	kij	kji	ikj	ijk	jki	jik
Access to A and U	row	column	row	column	column	column
Access to L	---------	column	---------	row	column	row
Computation of U	row	row	row	row	column	column
Computation of L	column	column	row	row	column	column
Vector operation	SAXPY	SAXPY	GAXPY	DOT	GAXPY	DOT
Vector length	2/3 n	2/3 n	2/3 n	n/3	2/3 n	n/3

Vector length = average of occuring vector lengths

Optimal form depends on storage of matrices and vector length.

How to Choose Blocks in L, resp. U Satisfying $LU = A$

$$\begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} =$$

$$= \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} & L_{11}U_{13} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} & L_{21}U_{13} + L_{22}U_{23} \\ L_{31}U_{11} & L_{31}U_{12} + L_{32}U_{22} & * \end{pmatrix}$$

How to Choose Blocks in L, resp. U Satisfying $LU = A$

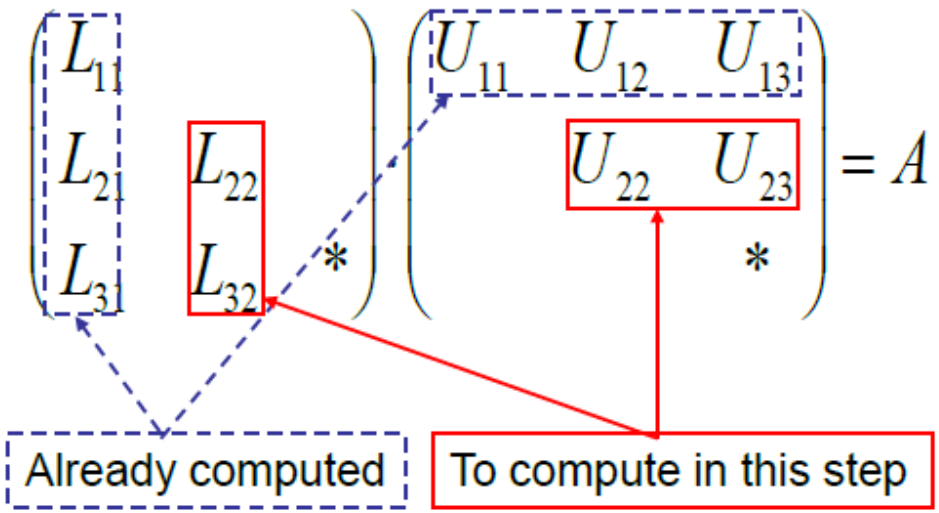
$$\begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} =$$

$$= \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} & L_{11}U_{13} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} & L_{21}U_{13} + L_{22}U_{23} \\ L_{31}U_{11} & L_{31}U_{12} + L_{32}U_{22} & * \end{pmatrix}$$

Different ways of computing L and U depending on

- start (assume first entry/row/column of L/U as given)
- how to compute new entry/row/column of L/U
- update of block structure of L/U by grouping in
  - known blocks
  - blocks newly to compute
  - blocks to be computed later

Crout Form



$$\begin{pmatrix} L_{22}U_{22} & L_{22}U_{23} \\ L_{32}U_{22} & * \end{pmatrix} \overset{!}{=} \begin{pmatrix} A_{22} - L_{21}U_{12} & A_{23} - L_{21}U_{13} \\ A_{32} - L_{31}U_{12} & * \end{pmatrix} = \begin{pmatrix} \hat{A}_{22} & \hat{A}_{23} \\ \hat{A}_{32} & * \end{pmatrix}$$

Leads to two subproblems in ▭ and in ▭

Crout Form (cont.)

1. Solve

$$\begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} \cdot U_{22} = \begin{pmatrix} \hat{A}_{22} \\ \hat{A}_{32} \end{pmatrix}$$

by small LU-decomposition of the modified part of $A \rightarrow L_{22}, L_{32}$, and U_{22}.

2. Solve

$$L_{22} \cdot U_{23} = \hat{A}_{23}$$

by solving small triangular systems of equations in $L_{22} \rightarrow U_{23}$.

Crout Form (cont.)

1. Solve

$$\begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} \cdot U_{22} = \begin{pmatrix} \hat{A}_{22} \\ \hat{A}_{32} \end{pmatrix}$$

   by small LU-decomposition of the modified part of $A \rightarrow L_{22}, L_{32}$, and U_{22}.

2. Solve

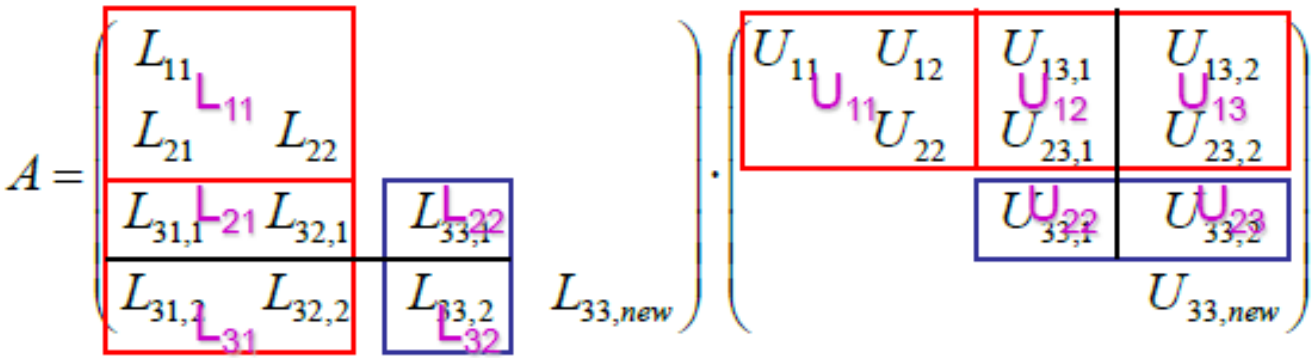
$$L_{22} \cdot U_{23} = \hat{A}_{23}$$

   by solving small triangular systems of equations in $L_{22} \rightarrow U_{23}$.

Initial steps:

$$L_{11} U_{11} = A_{11}, \quad \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} U_{11} = \begin{pmatrix} A_{21} \\ A_{31} \end{pmatrix}, \quad L_{11} (U_{12} \;\; U_{13}) = (A_{12} \;\; A_{13})$$

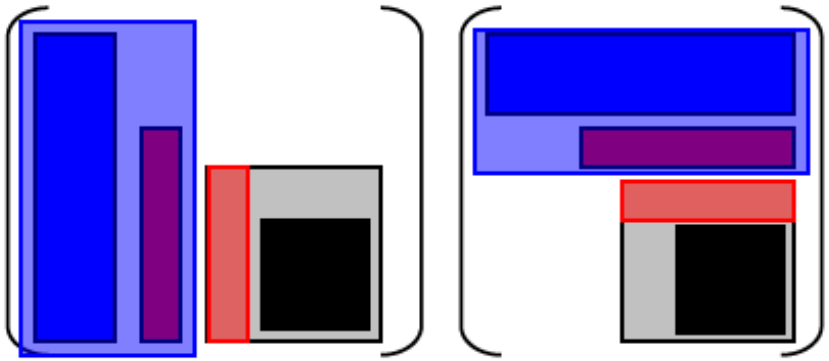
New Partitioning



- Combine already computed parts from second column of L and second row of U into first column of L and first row of U.
- Split the until now ignored parts L_{33} and U_{33} into new columns/rows.
- Repeat this overall procedure until L and U are fully computed.

Block Structure

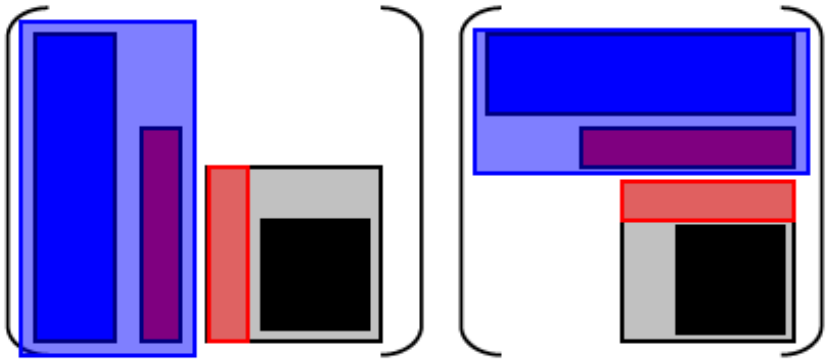
Intermediate block structure:



Solve for red blocks.

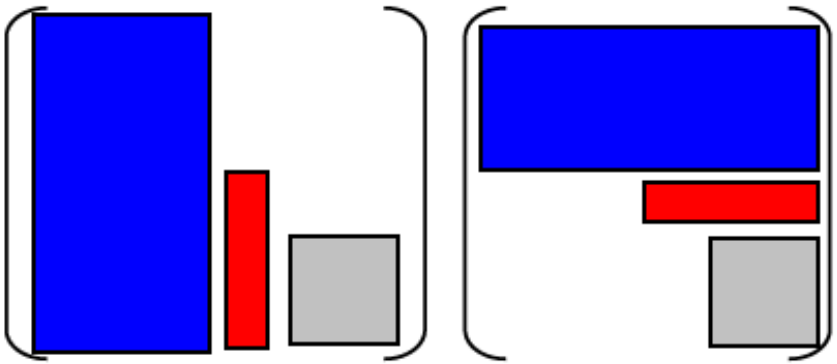
Block Structure

Intermediate block structure:



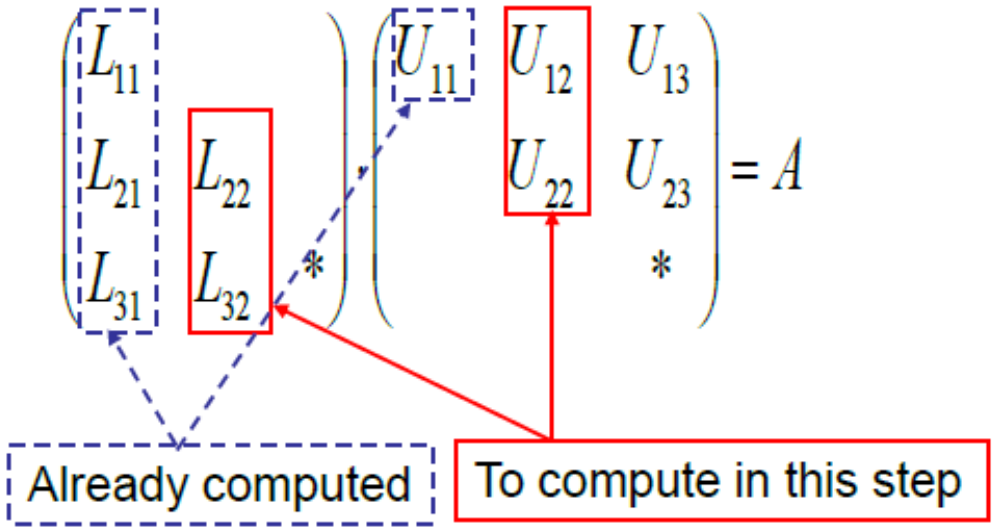
Solve for red blocks.

Reconfigure the block structure:

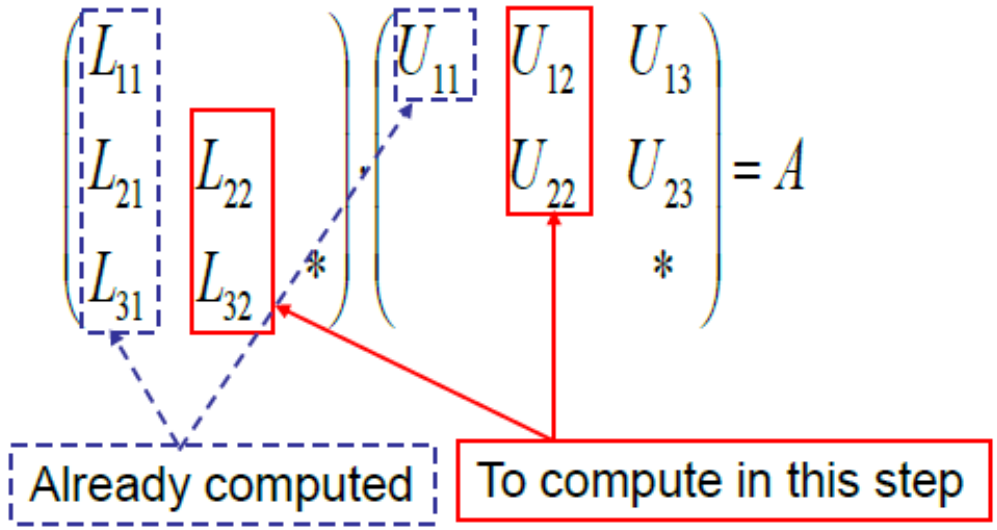


Repeat until done.

Left Looking GE



Left Looking GE

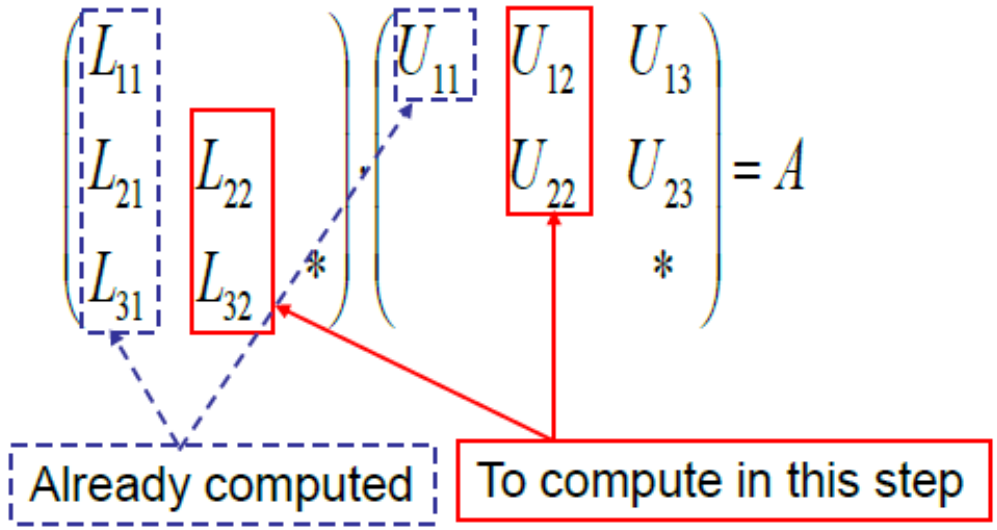


- Solve $L_{11} U_{12} = A_{12}$ by a couple of parallel triangular solves and

$$\begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22} = \begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} - \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} U_{12} =: \begin{pmatrix} \hat{A}_{22} \\ \hat{A}_{32} \end{pmatrix}$$

update part of A and perform small LU-decompostion.

Left Looking GE



- Solve $L_{11}U_{12} = A_{12}$ by a couple of parallel triangular solves and

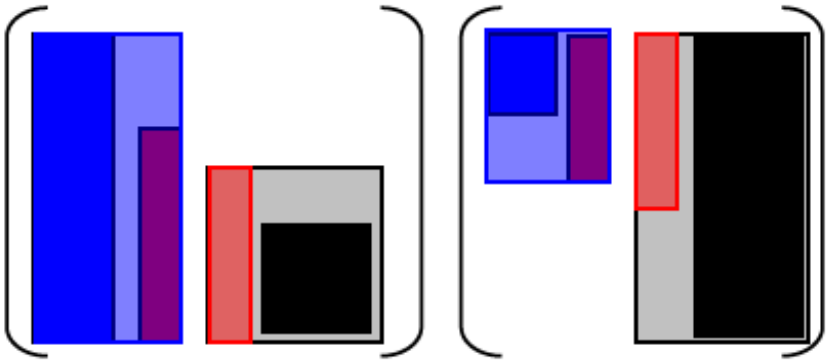
$$\begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22} = \begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} - \begin{pmatrix} L_{21} \\ L_{31} \end{pmatrix} U_{12} =: \begin{pmatrix} \hat{A}_{22} \\ \hat{A}_{32} \end{pmatrix}$$

  update part of A and perform small LU-decompostion.

- Reorder blocks and repeat until ready. Start: $L_{11}U_{11} = A_{11}$, $L_{21}U_{11} = A_{21}$, and $L_{31}U_{11} = A_{31}$.

Block Structure

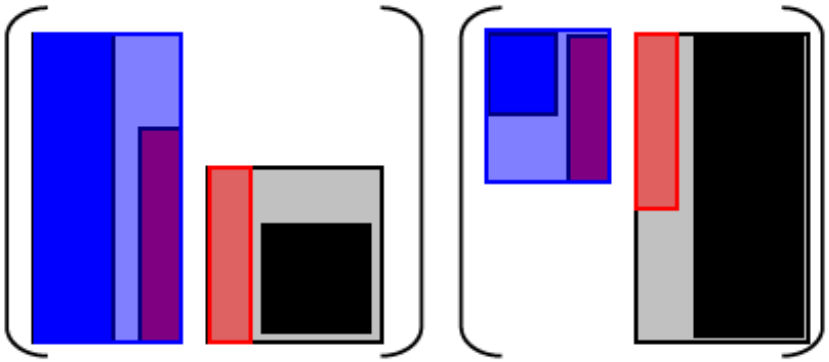
Intermediate block structure:



Solve for red blocks.

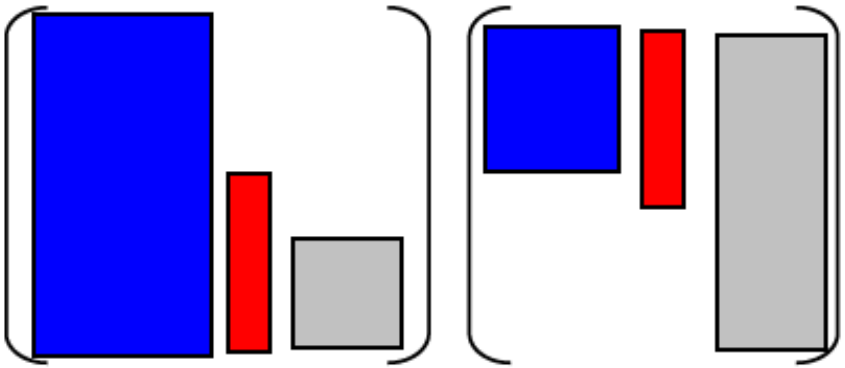
Block Structure

Intermediate block structure:



Solve for red blocks.

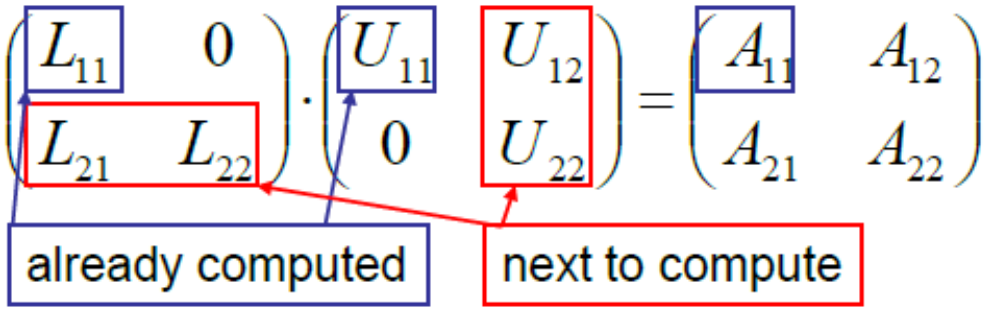
Reconfigure the block structure:



Repeat until done.

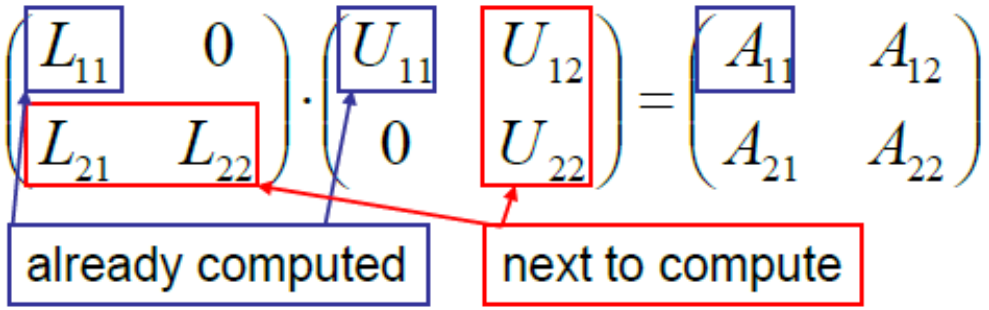
Right Looking GE

New blocking:



Right Looking GE

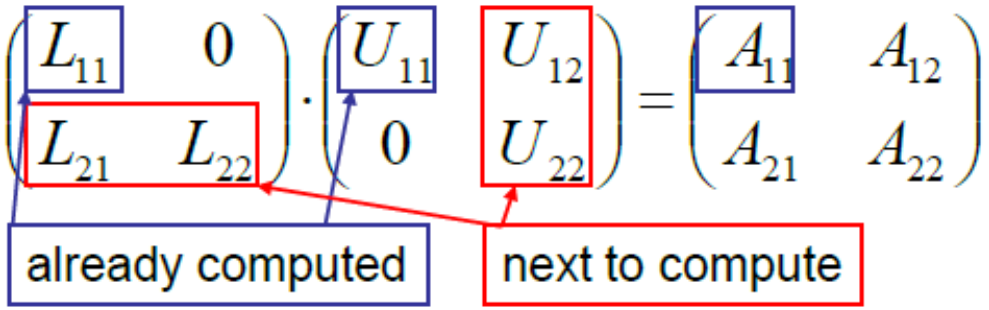
New blocking:



- Start with $L_{11}U_{11} = A_{11}$ (small LU-decomposition).

Right Looking GE

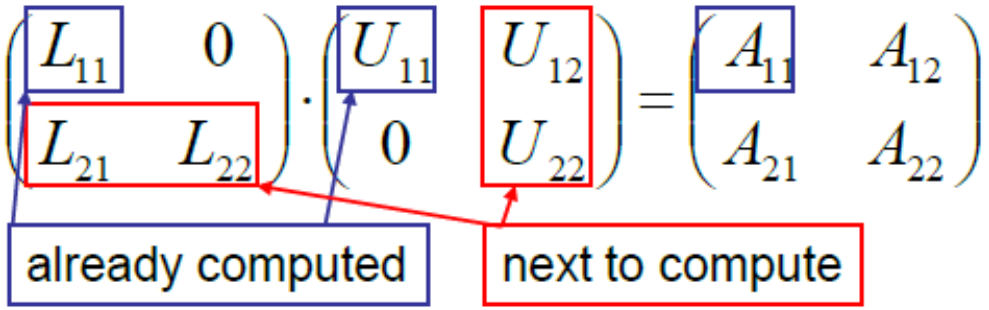
New blocking:



- Start with $L_{11}U_{11} = A_{11}$ (small LU-decomposition).
- Equations $L_{21}U_{11} = A_{21}$ and $L_{11}U_{12} = A_{12}$ by triangular solves gives L_{21} and U_{12}.

Right Looking GE

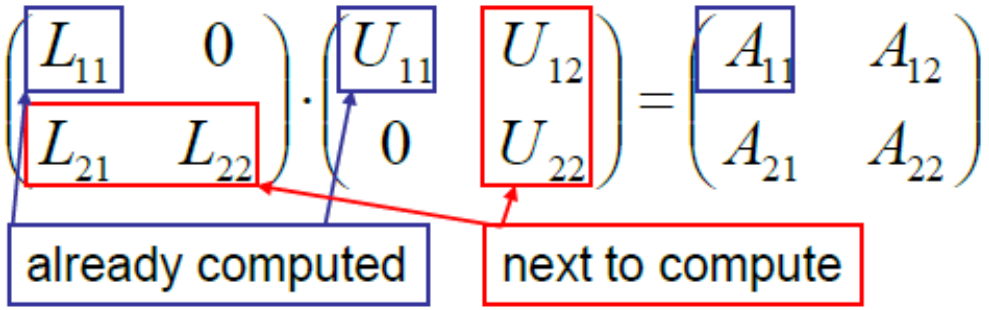
New blocking:



- Start with $L_{11}U_{11} = A_{11}$ (small LU-decomposition).
- Equations $L_{21}U_{11} = A_{21}$ and $L_{11}U_{12} = A_{12}$ by triangular solves gives L_{21} and U_{12}.
- It remains $L_{22}U_{22} = A_{22} - L_{21}U_{12} = \hat{A}_{22}$

Right Looking GE

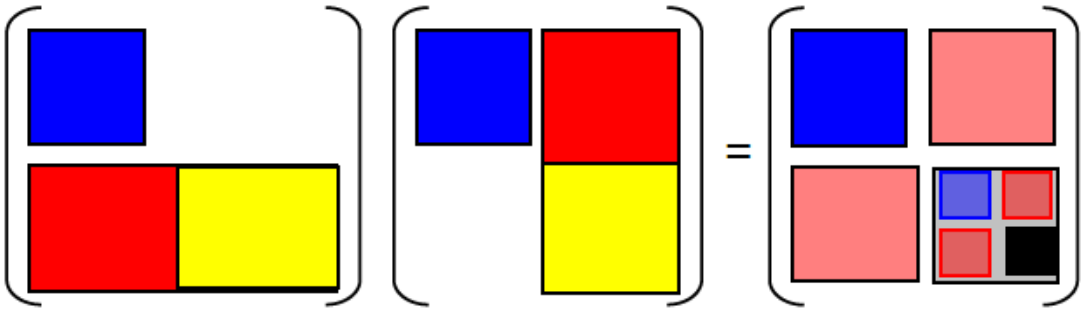
New blocking:



- Start with $L_{11}U_{11} = A_{11}$ (small LU-decomposition).
- Equations $L_{21}U_{11} = A_{21}$ and $L_{11}U_{12} = A_{12}$ by triangular solves gives L_{21} and U_{12}.
- It remains $L_{22}U_{22} = A_{22} - L_{21}U_{12} = \hat{A}_{22}$
- To compute the LU-decomposition of modified A_{22} repeat 2×2-blocking for A_{22} and apply recursively.

Block Structure

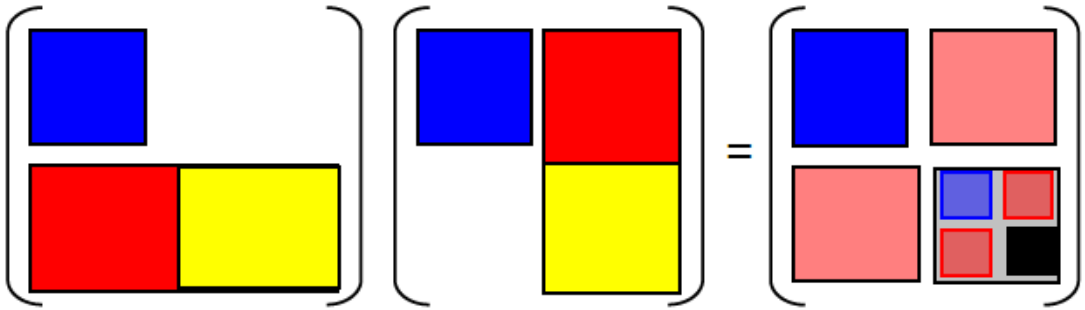
Intermediate block structure:



Solve for blue and both red blocks.

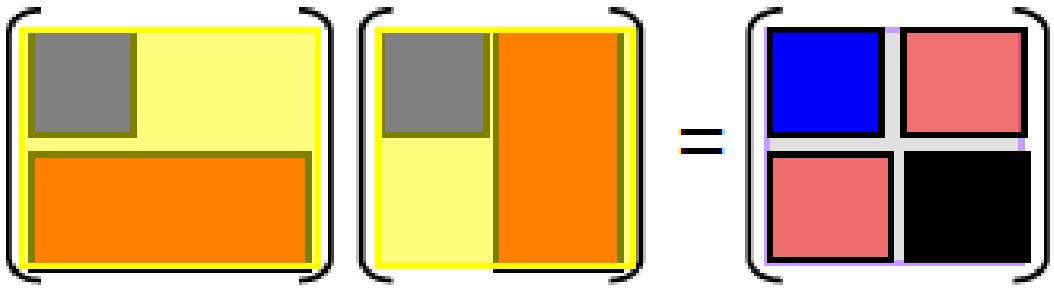
Block Structure

Intermediate block structure:



Solve for blue and both red blocks.

Reconfigure the block structure:



Repeat until done.

Comparison and Overview

- In comparison, all methods
  - have nearly same efficiency in parallel
  - but better performance (in sequential or parallel) than the unblocked variants because they are based on BLAS-3.

Comparison and Overview

- In comparison, all methods
  - have nearly same efficiency in parallel
  - but better performance (in sequential or parallel) than the unblocked variants because they are based on BLAS-3.
- Elementary steps of all blocking methods:
  - Matrix-Matrix product and sum (easy to parallelize)
  - Couple of triangular solves (easy to parallelize)
  - Small LU-decomposition (parallelizable for long rows)

Comparison and Overview

- In comparison, all methods
  - have nearly same efficiency in parallel
  - but better performance (in sequential or parallel) than the unblocked variants because they are based on BLAS-3.
- Elementary steps of all blocking methods:
  - Matrix-Matrix product and sum (easy to parallelize)
  - Couple of triangular solves (easy to parallelize)
  - Small LU-decomposition (parallelizable for long rows)
- Crout and right looking slightly better because more flops in matrix-updates and less triangular solves respectively *LU*-decompositions.

3.3. *QR*-Decomposition with Householder Matrices

3.3.1. QR-decomposition

- Gaussian elimination $\rightarrow$ *LU*-decomposition: sometimes numerically not stable, over/underdetermined systems

3.4. *QR*-Decomposition with Householder Matrices

3.4.1. QR-decomposition

- Gaussian elimination $\rightarrow$ *LU*-decomposition: sometimes numerically not stable, over/underdetermined systems
- Improvement:

  QR-decomposition $A = QR$ with Q orthogonal, R triangular,
  Solve linear system Ax=b numerically stable via

  $$b = Ax = QRx \Leftrightarrow Rx = Q^T b$$

  by cheap matrix-vector multiplication and triangular solve.

Overdetermined Systems

- $Ax = b$ with

  A being $m \times n$ matrix, $n \ll m$

  x vector of length n

  b vector of length m

Overdetermined Systems

- $Ax = b$ with

  A being $m \times n$ matrix, $n \ll m$

  x vector of length n

  b vector of length m

- Best **approximate** solution by solving minimization

$$\min_x \|Ax - b\|_2^2 = \min_x (x^T A^T Ax - 2x^T A^T b + b^T b)$$

- Gradient equal zero $\Leftrightarrow A^T Ax = A^T b$ (normal equations)

Overdetermined Systems

- $Ax = b$ with

  A being $m \times n$ matrix, $n \ll m$

  x vector of length n

  b vector of length m

- Best **approximate** solution by solving minimization

$$\min_x \|Ax - b\|_2^2 = \min_x (x^T A^T Ax - 2x^T A^T b + b^T b)$$

- Gradient equal zero $\Leftrightarrow A^T Ax = A^T b$ (normal equations)

- Solution by considering linear system $A^T A$, but condition number worse:

$$\text{cond}(A^T A) = \text{cond}(A)^2$$

Advantage of QR-Decomposition

$$A = QR, \quad R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad \text{cond}(R_1) = \text{cond}(A), \quad \hat{b} = Q^T b = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix}$$

$$A^T Ax = A^T b \Leftrightarrow (QR)^T (QR)x = (QR)^T b \Leftrightarrow$$

$$R^T Rx = R^T (Q^T b) \Leftrightarrow (R_1^T \ 0) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} x = (R_1^T \ 0)\hat{b} \Leftrightarrow$$

$$R_1^T R_1 x = (R_1^T \ 0) \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} \Leftrightarrow R_1^T R_1 x = R_1^T \hat{b}_1 \Leftrightarrow R_1 x = \hat{b}_1$$

Advantage of QR-Decomposition

$$A = QR, \quad R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad \text{cond}(R_1) = \text{cond}(A), \quad \hat{b} = Q^T b = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix}$$

$$A^T Ax = A^T b \Leftrightarrow (QR)^T (QR)x = (QR)^T b \Leftrightarrow$$

$$R^T Rx = R^T (Q^T b) \Leftrightarrow (R_1^T \ 0) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} x = (R_1^T \ 0)\hat{b} \Leftrightarrow$$

$$R_1^T R_1 x = (R_1^T \ 0) \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} \Leftrightarrow R_1^T R_1 x = R_1^T \hat{b}_1 \Leftrightarrow R_1 x = \hat{b}_1$$

- Instead of solving the normal equations we only have to consider the triangular system in R_1 .
- Cheap and better condition number.

3.4.2. Householder Method

- Define special orthogonal and simple matrices H called Householder matrices (compare Givens):

$$u \in \mathbb{R}^n, \ \|u\|_2 = 1 : \quad H = I - 2uu^T$$

3.4.3. Householder Method

- Define special orthogonal and simple matrices H called Householder matrices (compare Givens):

$$u \in \mathbb{R}^n, \ \|u\|_2 = 1 : \quad H = I - 2uu^T$$

- H as rank-1 perturbation of the identity is symmetric, idempotent and orthogonal:

$$H^T = I - 2uu^T = H$$

$$H^T H = H^2 = (I-2uu^T)(I-2uu^T) = I-2uu^T-2uu^T+4u\underbrace{u^T u}_{=1}u^T = I$$

3.4.4. Householder Method

- Define special orthogonal and simple matrices H called Householder matrices (compare Givens):

$$u \in \mathbb{R}^n, \ \|u\|_2 = 1 : \quad H = I - 2uu^T$$

- H as rank-1 perturbation of the identity is symmetric, idempotent and orthogonal:

$$H^T = I - 2uu^T = H$$

$$H^T H = H^2 = (I-2uu^T)(I-2uu^T) = I-2uu^T-2uu^T+4u\underbrace{u^T u}_{=1}u^T = I$$

- For complex problems:
orthogonal $\rightarrow$ unitary, symmetric $\rightarrow$ hermitian

Householder Method (cont.)

- Use H_1 with appropriate vector u_1 to eliminate first column of A

$$H_1 A = (I - 2u_1 u_1^T)(a_1 \ \cdots \ a_m) = (a_1 - 2(u_1^T a_1)u_1 \ \cdots \ *) = \begin{pmatrix} \alpha & * \\ 0 & * \\ \vdots & \vdots \\ 0 & * \end{pmatrix}$$

Householder Method (cont.)

- Use H_1 with appropriate vector u_1 to eliminate first column of A

$$H_1 A = (I - 2u_1 u_1^T)(a_1 \ \cdots \ a_m) = (a_1 - 2(u_1^T a_1)u_1 \ \cdots \ *) = \begin{pmatrix} \alpha & * \\ 0 & * \\ \vdots & \vdots \\ 0 & * \end{pmatrix}$$

- To satisfy this equation we have to find a vector u_1 of length 1 with

$$a_1 - 2(u_1^T a_1)u_1 = \alpha e_1$$

Householder Method (cont.)

- Use H_1 with appropriate vector u_1 to eliminate first column of A

$$H_1 A = (I - 2u_1 u_1^T)(a_1 \ \cdots \ a_m) = (a_1 - 2(u_1^T a_1)u_1 \ \cdots \ *) = \begin{pmatrix} \alpha & * \\ 0 & * \\ \vdots & \vdots \\ 0 & * \end{pmatrix}$$

- To satisfy this equation we have to find a vector u_1 of length 1 with

$$a_1 - 2(u_1^T a_1)u_1 = \alpha e_1$$

- Because H_1 is orthogonal it holds:

$$\|H_1 a_1\|_2 = \|a_1\|_2 = \|\alpha e_1\|_2 = |\alpha| \Rightarrow \alpha = \pm\|a_1\|_2, \text{ e.g. } \alpha = \|a_1\|_2$$

$$u_1 = \frac{a_1 - \|a_1\|_2 e_1}{2(u_1^T a_1)} = \frac{a_1 - \|a_1\|_2 e_1}{\|a_1 - \|a_1\|_2 e_1\|_2}$$

Householder Method (cont. 2)

- Repeat for all columns of A

$$H_1 A = H_1 A_1 = (I - 2u_1 u_1^T)A = \left(\begin{array}{c|ccc} \|a_1\|_2 & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{array}\right)$$

Householder Method (cont. 2)

- Repeat for all columns of A

$$H_1 A = H_1 A_1 = (I - 2u_1 u_1^T)A = \left(\begin{array}{c|ccc} \|a_1\|_2 & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{array}\right)$$

- Apply the same procedure on A_2, $(n-1) \times (m-1)$ matrix.

$$\tilde{H}_2 A_2 = (I - 2\tilde{u}_2 \tilde{u}_2^T)A_2 = \left(\begin{array}{c|ccc} \alpha_2 & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A_3 & \\ 0 & & & \end{array}\right)$$

Householder Method (cont. 2)

- Repeat for all columns of A

$$H_1 A = H_1 A_1 = (I - 2u_1 u_1^T)A = \left(\begin{array}{c|ccc} \|a_1\|_2 & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{array}\right)$$

- Apply the same procedure on A_2, $(n-1) \times (m-1)$ matrix.

$$\tilde{H}_2 A_2 = (I - 2\tilde{u}_2 \tilde{u}_2^T)A_2 = \left(\begin{array}{c|ccc} \alpha_2 & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A_3 & \\ 0 & & & \end{array}\right)$$

- Extend

$$u_2 := \begin{pmatrix} 0 \\ \tilde{u}_2 \end{pmatrix}, \qquad H_2 := I - 2u_2 u_2^T = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{H}_2 & \\ 0 & & & \end{pmatrix}$$

Householder Method (cont. 3)

- For column $1, 2, \ldots, m$ this gives Householder matrices $H_1, \ldots, H_m$ with

$$\underbrace{H_m \cdots H_2 H_1}_{= Q^T} \cdot A = H_m \cdots H_3 \cdot \left(\begin{array}{ccc|ccc} \alpha_1 & * & * & \cdots & * \\ 0 & \alpha_2 & * & \cdots & * \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & A_3 & \\ 0 & 0 & & & \end{array}\right) = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} =: R$$

Householder Method (cont. 3)

- For column $1, 2, \ldots, m$ this gives Householder matrices $H_1, \ldots, H_m$ with

$$\underbrace{H_m \cdots H_2 H_1}_{= Q^T} \cdot A = H_m \cdots H_3 \cdot \left(\begin{array}{cc|ccc} \alpha_1 & * & * & \cdots & * \\ 0 & \alpha_2 & * & \cdots & * \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & A_3 & \\ 0 & 0 & & & \end{array}\right) = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} =: R$$

- Hence:

$$A = QR, \qquad Q := (H_m \cdots H_2 H_1)^T = H_1 H_2 \cdots H_m$$

Householder Method (cont. 3)

- For column $1, 2, \ldots, m$ this gives Householder matrices $H_1, \ldots, H_m$ with

$$\underbrace{H_m \cdots H_2 H_1}_{= Q^T} \cdot A = H_m \cdots H_3 \cdot \left(\begin{array}{ccc|ccc} \alpha_1 & * & * & \cdots & * \\ 0 & \alpha_2 & * & \cdots & * \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & A_3 & \\ 0 & 0 & & & \end{array}\right) = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} =: R$$

- Hence:

$$A = QR, \qquad Q := (H_m \cdots H_2 H_1)^T = H_1 H_2 \cdots H_m$$

- Remark: for $m = n$: $H_1, \ldots, H_{m-1}$ is enough, because last column is scalar.

3.4.5. Householder Method in Parallel - Blockwise

Idea: work again blockwise.

- In a first step compute u_1 and the application of H_1 on the first k columns of A. Do not compute $H_1 A$ fully!

3.4.6. Householder Method in Parallel - Blockwise

Idea: work again blockwise.

- In a first step compute u_1 and the application of H_1 on the first k columns of A. Do not compute $H_1 A$ fully!
- Then compute $u_2, \ldots, u_k$ and the application of $H_1 \ldots H_k$ on the first columns of A.

$$H_k \cdots H_1 (A_1 \quad A_2) = (H_k \cdots H_1 A_1 \quad (H_k \cdots H_1) A_2) = (A_1^{(k)} \quad VA_2)$$

3.4.7. Householder Method in Parallel - Blockwise

Idea: work again blockwise.

- In a first step compute u_1 and the application of H_1 on the first k columns of A. Do not compute $H_1 A$ fully!
- Then compute $u_2, \ldots, u_k$ and the application of $H_1 \ldots H_k$ on the first columns of A.

$$H_k \cdots H_1 (A_1 \quad A_2) = (H_k \cdots H_1 A_1 \quad (H_k \cdots H_1) A_2) = (A_1^{(k)} \quad VA_2)$$

- Still to compute: VA_2.
- How can we take advantage of parallelism in this computation? $\rightarrow$ represent V in special form that allows fast and parallel evaluation of VA_2.

Property of Householder matrices

Theorem 3: For Householder matrices $H_k, \ldots, H_i$ it holds

$$H_k \cdots H_i = (I - 2u_k u_k^T) \cdots (I - 2u_i u_i^T) = I - \underbrace{(u_k \quad \cdots \quad u_i)}_{=: Y} T_i \begin{pmatrix} u_k^T \\ \vdots \\ u_i^T \end{pmatrix}$$

with T_i being upper triangular.

Property of Householder matrices

Theorem 3: For Householder matrices $H_k, \ldots, H_i$ it holds

$$H_k \cdots H_i = (I - 2u_k u_k^T) \cdots (I - 2u_i u_i^T) = I - \underbrace{(u_k \quad \cdots \quad u_i)}_{=: Y} T_i \begin{pmatrix} u_k^T \\ \vdots \\ u_i^T \end{pmatrix}$$

with T_i being upper triangular.

Proof by induction:

Representation obviously fulfilled for one Householder mtx $i = k$.

Assume, representation holds for $H_k, \ldots, H_i$. Then ... (next slide)

Property of Householder matrices (cont.)

$$\left[(I - 2u_k u_k^T) \cdots (I - 2u_i u_i^T)\right](I - 2u_{i-1} u_{i-1}^T) =$$

$$= \left[I - (u_k \quad \cdots \quad u_i) T_i \begin{pmatrix} u_k^T \\ \vdots \\ u_i^T \end{pmatrix}\right] \cdot (I - 2u_{i-1} u_{i-1}^T) =$$

$$= I - 2u_{i-1} u_{i-1}^T - (u_k \quad \cdots \quad u_i) T_i \begin{pmatrix} u_k^T \\ \vdots \\ u_i^T \end{pmatrix} + 2(u_k \quad \cdots \quad u_i) \underbrace{T_i \begin{pmatrix} u_k^T u_{i-1} \\ \vdots \\ u_i^T u_{i-1} \end{pmatrix}}_{=: y} u_{i-1}^T =$$

$$= I - (u_k \quad \cdots \quad u_i \quad u_{i-1}) \cdot \begin{pmatrix} T_i & -2y \\ 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} u_k^T \\ \vdots \\ u_i^T \\ u_{i-1}^T \end{pmatrix}$$

Algorithm for parallel Householder

Computation of $H_k \cdots H_i A = V_{ki} A = (I - YTY^T)A$ in the form

$$V_{ki} A = V_{ki}(A_1 \quad A_2) = (* \quad V_{ki} A_2)$$

and

$$V_{ki} A_2 = (I - YTY^T)A_2 = A_2 - Y[T(Y^T A_2)]$$

Algorithm for parallel Householder

Computation of $H_k \cdots H_i A = V_{ki} A = (I - YTY^T)A$ in the form

$$V_{ki} A = V_{ki}(A_1 \quad A_2) = (* \quad V_{ki} A_2)$$

and

$$V_{ki} A_2 = (I - YTY^T)A_2 = A_2 - Y[T(Y^T A_2)]$$

Algorithm:

- Compute u_1 and $H_1 A_1$; u_2 and $H_2 A_1$; ...; u_k and $H_k A_1$ (sequential)
- Compute Y and VA_2 (parallel)
- Repeat with indices $k+1, \ldots, 2k$; $2k+1, \ldots, 3k$; ...

Algorithm for parallel Householder

Computation of $H_k \cdots H_i A = V_{ki} A = (I - YTY^T)A$ in the form

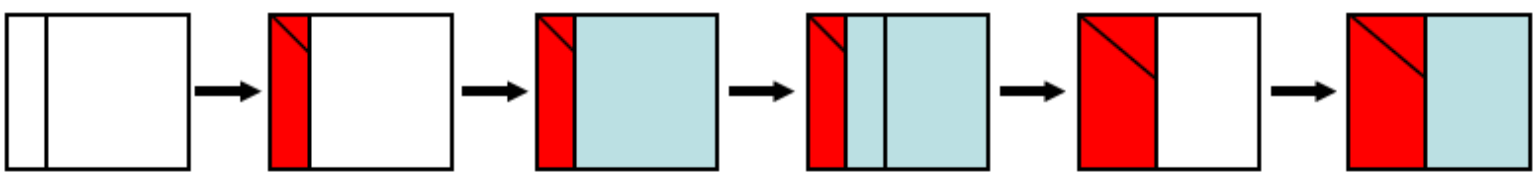
$$V_{ki} A = V_{ki}(A_1 \quad A_2) = (* \quad V_{ki} A_2)$$

and

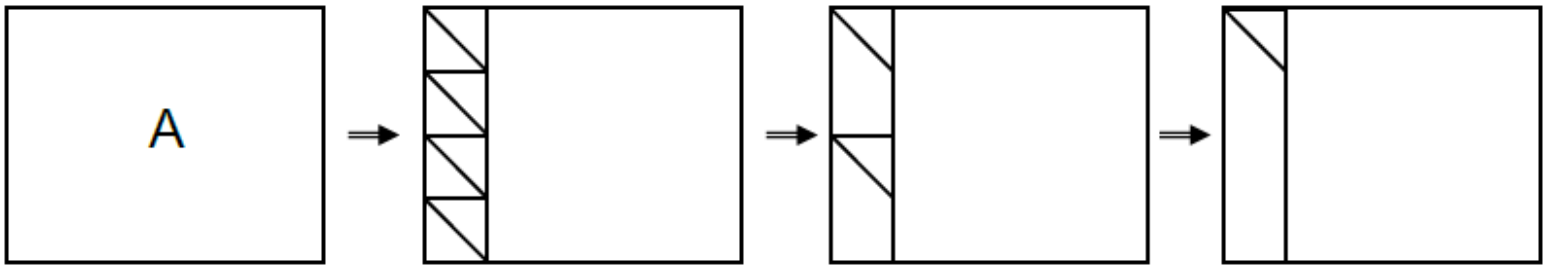
$$V_{ki} A_2 = (I - YTY^T)A_2 = A_2 - Y[T(Y^T A_2)]$$

Algorithm:

- Compute u_1 and $H_1 A_1$; u_2 and $H_2 A_1$; ...; u_k and $H_k A_1$ (sequential)
- Compute Y and VA_2 (parallel)
- Repeat with indices $k+1, \ldots, 2k$; $2k+1, \ldots, 3k$; ...



Communication Avoiding QR



- First level: Four independent QR-factorisations

↓

- Second level: Two independent reduced QR-factorisations

↓

- Last level: One reduced QR-factorisation

Tall Skinny QR

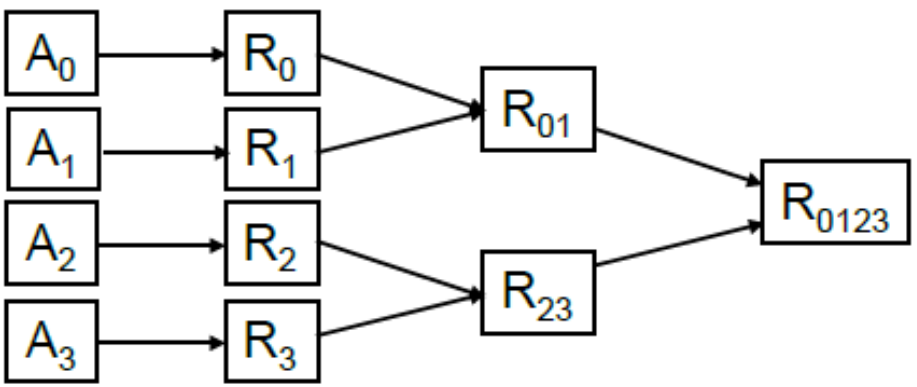
$$A = \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} Q_0 R_0 \\ Q_1 R_1 \\ Q_2 R_2 \\ Q_3 R_3 \end{pmatrix} = \begin{pmatrix} Q_0 & & & \\ & Q_1 & & \\ & & Q_2 & \\ & & & Q_3 \end{pmatrix} \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix}$$

$$\begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix} = \left(\begin{pmatrix} R_0 \\ R_1 \end{pmatrix} \right) = \begin{pmatrix} Q_{01} R_{01} \\ Q_{23} R_{23} \end{pmatrix} = \begin{pmatrix} Q_{01} & \\ & Q_{23} \end{pmatrix} \begin{pmatrix} R_{01} \\ R_{23} \end{pmatrix}$$

$$\begin{pmatrix} R_{01} \\ R_{23} \end{pmatrix} = Q_{0123} R_{0123}$$

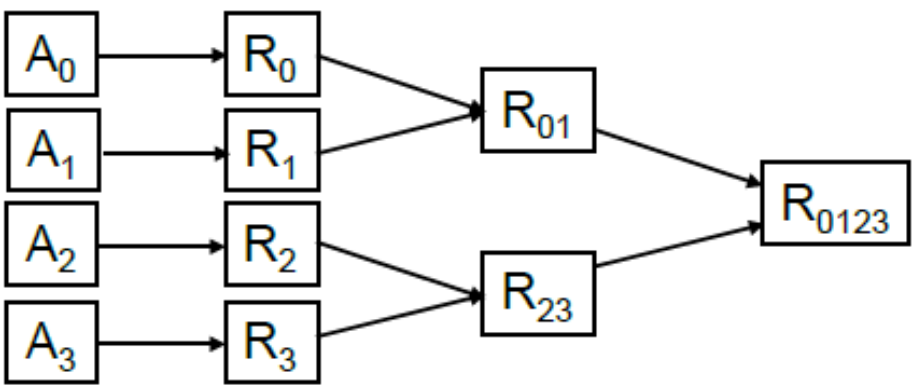
Tall Skinny QR (cont.)

$$A = \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \left\{ \begin{pmatrix} Q_0 & & & \\ & Q_1 & & \\ & & Q_2 & \\ & & & Q_3 \end{pmatrix} \cdot \begin{pmatrix} Q_{01} & \\ & Q_{23} \end{pmatrix} \cdot Q_{0123} \right\} \cdot R_{0123}$$



Tall Skinny QR (cont.)

$$A = \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \left\{ \begin{pmatrix} Q_0 & & & \\ & Q_1 & & \\ & & Q_2 & \\ & & & Q_3 \end{pmatrix} \cdot \begin{pmatrix} Q_{01} & \\ & Q_{23} \end{pmatrix} \cdot Q_{0123} \right\} \cdot R_{0123}$$



Advantage:
Messages in $\mathcal{O}(\log(P))$ compared to $\mathcal{O}(2n\log(P))$ for ScaLAPACK.

Cholesky QR Decomp. for Tall Skinny A

$$A^T A = LL^T$$

$$Q^T := L^{-1} A^T$$

Cholesky QR Decomp. for Tall Skinny A

$$A^T A = LL^T$$

$$Q^T := L^{-1} A^T$$

$$Q^T Q = L^{-1} A^T A L^{-T} = I$$

Cholesky QR Decomp. for Tall Skinny A

$$A^T A = LL^T$$

$$Q^T := L^{-1} A^T$$

$$Q^T Q = L^{-1} A^T A L^{-T} = I$$

$$A = QL^T = QR = \begin{pmatrix} Q & Q^\perp \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Cholesky QR Decomp. for Tall Skinny A

$$A^T A = LL^T$$

$$Q^T := L^{-1} A^T$$

$$Q^T Q = L^{-1} A^T A L^{-T} = I$$

$$A = QL^T = QR = \begin{pmatrix} Q & Q^\perp \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Advantage: Computation of $A^T A$ fully parallel, only small Cholesky decomposition L.

Cholesky QR Decomp. for Tall Skinny A

$$A^T A = LL^T$$

$$Q^T := L^{-1} A^T$$

$$Q^T Q = L^{-1} A^T A L^{-T} = I$$

$$A = QL^T = QR = \begin{pmatrix} Q & Q^\perp \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Advantage: Computation of $A^T A$ fully parallel, only small Cholesky decomposition L.

Disadvantage: Numerical stability.

Cholesky QR Decomp. for Tall Skinny A

$$A^T A = LL^T$$

$$Q^T := L^{-1} A^T$$

$$Q^T Q = L^{-1} A^T A L^{-T} = I$$

$$A = QL^T = QR = \begin{pmatrix} Q & Q^\perp \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

Advantage: Computation of $A^T A$ fully parallel, only small Cholesky decomposition L.

Disadvantage: Numerical stability.

Compromise:
Use Cholesky-QR only for well-conditioned A.