Introduction to Languages for Scientific Computing

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

LU Factorization $[L, U] = LU(A)$

Input: $A \in \mathbb{R}^{n \times n}$.

Output: $L \in \mathbb{R}^{n \times n}$; L is lower triangular with unit diagonal,
$U \in \mathbb{R}^{n \times n}$; U is upper triangular.

Algorithm: $[L, U] := \mathsf{LU}(A, b)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$, $L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $U \rightarrow \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right)$

where A_{TL}, L_{TL} and U_{TL} are 0×0

While $m(A_{TL}) < m(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right), \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) \rightarrow \ldots$$

where A_{11}, L_{11} and U_{11} are $b \times b$

$$U_{01} := L_{00}^{-1} A_{01}$$

$$[L_{11}, U_{11}] := \mathsf{LU}(A_{11} - L_{10} U_{01})$$

$$L_{21} := (A_{21} - L_{20} U_{01}) U_{11}^{-1}$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right), \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) \leftarrow \ldots$$

endWhile

LU — Matlab

- The function $m(M)$ returns the number of rows of M.
- Notice that the output matrices L and U can be stored in only one matrix, and in fact, they can overwrite A. In fact, at each iteration
  - U_{01} is computed from A_{01},
  - L_{11} and U_{11} are computed from A_{11}, and
  - L_{21} is computed from A_{21},

  and the contents of the input quadrants A_{01}, A_{11}, or A_{21} are not used in any subsequent iteration.
  This means that at each iteration, U_{01} can overwrite A_{01}, L_{11} and U_{11} can overwrite A_{11}, and L_{21} can overwrite A_{21}.
- In light of these consideration, the algorithm can be rewritten as in the next page.
- When L and U are stored in A, the diagonal of L—which consists of 1s—is not stored, and only implicitly defined. The functions $L(M)$ and $U(M)$ respectively extract L and U from M.
- Matlab's commands to extract a triangular portion of a matrix are `tril` and `triu`.

LU Factorization $A = LU(A)$

Input: $A \in \mathbb{R}^{n \times n}$.

Output: $L \in \mathbb{R}^{n \times n}$; L is lower triangular with unit diagonal,
$U \in \mathbb{R}^{n \times n}$; U is upper triangular.
L and U are stored in A

Algorithm: $A := \mathsf{LU}(A, b)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

While $m(A_{TL}) < m(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{01} := L(A_{00})^{-1} A_{01}$

$A_{11} := \mathsf{LU}(A_{11} - A_{10} A_{01})$

$A_{21} := (A_{21} - A_{20} A_{01}) U(A_{11})^{-1}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endWhile

LU — Matlab

1. Code up in Matlab the second algorithm for the LU factorization (without using `L` and `U`), aiming at readibility and elegance. **Let the code speak for itself!**
2. The prototype must be `LU( A, b )`, where `A` is the input matrix, and `b` is the block size.
3. For the recursive calls, use your own routine with block size = 1. `LU( *, 1)`.
4. Generate an informative plot to show the accuracy of the routines against problem size. How to measure accuracy? Export the plot as `accuracy.pdf`.
5. Generate another plot to show how the block size **b** affects the computation time. Consider both multiple problem sizes and multiple block sizes. Export the plot as `speed.pdf`.

Submission rules

- Individual assignment.
- Prepare three files: `LU.m`, `accuracy.pdf`, `speed.pdf`.
  Make sure to include your name in `LU.m`.
- Archive the files as `your_name.zip` or `your_name.tgz` and submit.
- Submission by email to `pauldj@aices.rwth-aachen.de`
- Email's subject: ‘‘`LSC-15 challenge2 <your last name>`’’
- **Deadline: Thursday, November 26, noon.**