# Parallel programming

# C++11 – assignment

Libor Bukata a Jan Dvořák

FAKULTA
ELEKTROTECHNICKÁ
ČVUT V PRAZE

# LU decomposition

- A decomposition of matrix **A** to lower and upper triangular matrices **L** and **U**, respectively.

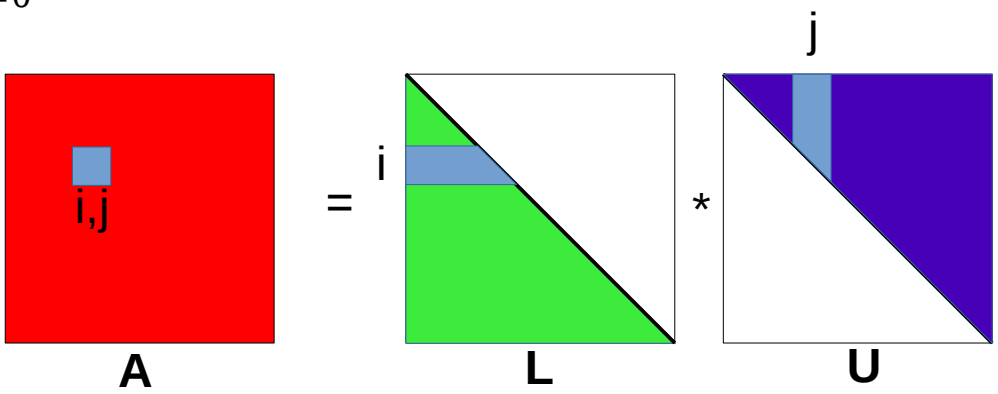
$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

$$A = LU$$

- **Applications:**
  - Useful to quickly resolve a linear system of equation if only the right-hand side is changed (Ax = b).
  - Simplex method may employ this decomposition to quickly update the inverse of basis.
  - Determinant can be readily calculated from **U** matrix.

# LU decomposition - derivation

$$a_{ij} = \sum_{r=0}^{n-1} l_{ir} u_{rj} \qquad \forall i, j \in \{0, \ldots, n-1\}$$




$$l_{ir} = \begin{cases} 0 \text{ if } r > i \\ 1 \text{ if } r = i \\ l_{ir} \text{ otherwise} \end{cases}$$

$$u_{rj} = \begin{cases} 0 \text{ if } r > j \\ u_{rj} \text{ otherwise} \end{cases}$$

$$a_{ij} = \sum_{r=0}^{n-1} l_{ir} u_{rj} = \sum_{r=0}^{i} l_{ir} u_{rj} = \sum_{r=0}^{i-1} l_{ir} u_{rj} + l_{ii} u_{ij} = \sum_{r=0}^{i-1} l_{ir} u_{rj} + 1\, u_{ij}$$

$$\Rightarrow \quad u_{ij} = a_{ij} - \sum_{r=0}^{i-1} l_{ir} u_{rj}$$

$$a_{ij} = \sum_{r=0}^{n-1} l_{ir} u_{rj} = \sum_{r=0}^{j} l_{ir} u_{rj} = \sum_{r=0}^{j-1} l_{ir} u_{rj} + l_{ij} u_{jj}$$

$$\Rightarrow \quad l_{ij} = \frac{1}{u_{jj}} \left( a_{ij} - \sum_{r=0}^{j-1} l_{ir} u_{rj} \right)$$

# LU decomposition - derivation

$$a_{ij}^{(k)} = a_{ij} - \sum_{r=0}^{k-1} l_{ir}\, u_{rj} \qquad \forall\, k \in \{0, \ldots, n-1\}$$

$$\Rightarrow \quad u_{ij} = a_{ij}^{(i)} \qquad l_{ij} = \frac{a_{ij}^{(j)}}{u_{jj}}$$

$$l_{ij} = \frac{1}{u_{jj}} \left( a_{ij} - \sum_{r=0}^{j-1} l_{ir}\, u_{rj} \right)$$

$$u_{ij} = a_{ij} - \sum_{r=0}^{i-1} l_{ir}\, u_{rj}$$

## Pseudocode:

**for** $k$ = 0 **to** n - 1 **do**
  **for** $j$ = $k$ **to** n - 1 **do**
    $u_{kj} = a_{kj}^{(k)}$
  $l_{kk} = 1$
  **for** $i$ = $k$ + 1 **to** n - 1 **do**
    $l_{ik} = a_{ik}^{(k)} / u_{kk}$
  **for** $i$ = $k$ + 1 **to** n - 1 **do**
    **for** $j$ = $k$ + 1 **to** n - 1 **do**
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}\, u_{kj}$

# LU decomposition - example

$$A=\begin{pmatrix}1 & 4 & 6\\ 2 & 10 & 17\\ 3 & 16 & 31\end{pmatrix}=\begin{pmatrix}1 & 0 & 0\\ 2 & 1 & 0\\ 3 & 2 & 1\end{pmatrix}\begin{pmatrix}1 & 4 & 6\\ 0 & 2 & 5\\ 0 & 0 & 3\end{pmatrix}$$

## iteration 0

$u_{00}=a_{00}^{(0)}=1$ $\quad l_{00}=1$

$u_{01}=a_{01}^{(0)}=4$ $\quad l_{10}=a_{10}^{(0)}/u_{00}=2/1=2$

$u_{02}=a_{02}^{(0)}=6$ $\quad l_{20}=a_{20}^{(0)}/u_{00}=3/1=3$

$a_{11}^{(1)}=a_{11}^{(0)}-l_{10}u_{01}=10-2*4=2$ $\quad a_{12}^{(1)}=a_{12}^{(0)}-l_{10}u_{02}=17-2*6=5$

$a_{21}^{(1)}=a_{21}^{(0)}-l_{20}u_{01}=16-3*4=4$ $\quad a_{22}^{(1)}=a_{22}^{(0)}-l_{20}u_{02}=31-3*6=13$

## iteration 1

$u_{11}=a_{11}^{(1)}=2$ $\quad l_{11}=1$

$u_{12}=a_{12}^{(1)}=5$ $\quad l_{21}=a_{21}^{(1)}/u_{11}=4/2=2$

$a_{22}^{(2)}=a_{22}^{(1)}-l_{21}u_{12}=13-2*5=3$

## iteration 2

$u_{22}=a_{22}^{(2)}=3$ $\quad l_{22}=1$

# LU assignment

- Use the **provided snippet of code** (needed for upload system)
  - ➔ reads test problems
  - ➔ measures runtime
  - ➔ prints the matrices **L**, **U**, and **A**.
- **Assignment:**
  - ➔ implement LU decomposition
  - ➔ parallelize the code using C++11 threads
  - ➔ upload your solution to Course Ware
- **Tricky issues:**
  - For large problems the performance is memory bound, therefore, it may happen that the achieved speedup is relatively low depending on scaling of the memory system. You can mitigate the issue by implementing a cache-friendly version (cache blocking technique).
  - In practise, a partial pivoting is used to have numerically stable results (avoid possible division by zero).
  - You may require to implement barrier by using e.g. atomics.