LU decomposition without pivoting: $A = LU$

$A \in \mathbb{R}^{n\times n}$

$L \in \mathbb{R}^{n\times n}$ unit lower triangular

$U \in \mathbb{R}^{n\times n}$ upper triangular

Theorem $A \in \mathbb{R}^{n\times n}$, then $\exists$ permutation matrix $P \in \mathbb{R}^{n\times n}$, $L, U \in \mathbb{R}^{n\times n}$:

$$PA = LU$$

Proof: (Sketch)

By induction over size n:

$n=2$ $\quad A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Case 1: $a \neq 0 \longrightarrow$ Theorem on dominant submatrices

$\Longrightarrow A = LU$, $P = I$ ✓

Case 2: $a = 0$, $c \neq 0$, $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

$$PA = \begin{bmatrix} c & d \\ 0 & b \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} c & d \\ 0 & b \end{bmatrix}}_{U}$$

Case 3: $a = c = 0$, $P = I$

$$PA = \begin{bmatrix} 0 & b \\ 0 & d \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{=L} \underbrace{\begin{bmatrix} 0 & b \\ 0 & d \end{bmatrix}}_{=U}$$

Induction step: Choose permutation matrix $P^{(1)}$ such that $P^{(1)}A$ has the element with largest absolute value amongst all elements in the 1st column on (1,1) spot.

.... assume a block LU-decomposition and show that it is well-defined

$$P^{1r} A = \left[\begin{array}{c|c} \alpha & w^T \\ \hline p & B \end{array}\right]$$ and use induction assumption on B ✓

and if all entries in 1st column are zero, using the assumption for smaller matrices → ✓

($\square$)

Recall: to solve linear system $Ax=b$

1.) LU decomposition $PA = LU$

2.) Forward substitution $Ly = Pb$

3.) Backward —"— $Ux = y$

then x solves $Ax = b$.

$Ax = b \Leftrightarrow$

$PAx = Pb \Leftrightarrow$

$LUx = Pb \Leftrightarrow$

$y = Ux$
$Ly = Pb$

---

§ 2.6 Computational cost

We count the number of elementary operations $(+, \cdot, /, -)$ as a measure of the cost of algorithms, ("flops")

Consider LU without permutations (for simplicity)

$$l_{ij} = \frac{1}{u_{jj}} \left\{ a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right\} \qquad \begin{array}{l} i = 2, 3, \dots, n \\ j = 1, \dots, i-1 \end{array} \qquad \begin{array}{l} j-1 \text{ mult} \\ j-1 \text{ add/sub} \\ +1 \text{ division} \end{array}$$

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad \begin{array}{l} i = 1, \dots, n \\ j = i, \dots, n \end{array} \qquad \begin{array}{l} i-1 \text{ mult} \\ i-1 \text{ add/subtr.} \end{array}$$

$$\sum_{i=2}^{n} \sum_{j=1}^{i-1} 2j-1 + \sum_{i=1}^{n} \sum_{j=i}^{n} 2i-2 = \frac{1}{6} n(n-1)(4n+1)$$

$$\sim \frac{2}{3} n^3 - \frac{1}{2} n^2$$

Forward/backward substitution:

forw: $Ly = Pb$

$$y_i = (Pb)_i - \sum_{j=1}^{i-1} l_{ij} \, y_j \qquad i = 2, 3, \dots, n$$ $i-1$ mult, $i-1$ subst.

$$\sum_{i=2}^{n} 2i-2 = 2\,\frac{n(n-1)}{2} = n(n-1) \sim n^2$$

backward: $\sim n^2$

overall cost: $\sim \frac{2}{3}n^3 + \frac{3}{2}n^2$

Let us assume we want to solve a system with k different right hand sides, i.e.:

$$A x_i = b_i \qquad i = 1, \dots, k$$

cost: $\underbrace{\frac{2}{3}n^3 - \frac{1}{2}n^2}_{\text{LU factorization}} + \underbrace{2kn^2}_{k \text{ forw/back subst.}}$

Is it better to compute A^{-1} and then $A^{-1}b_i$ $i = 1, \dots, k$?

cost of computing A^{-1}: $AA^{-1} = I$, i.e. to get a column of A^{-1}, solve a system with rhs vector $e_i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$

cost: $\underbrace{\frac{2}{3}n^3 - \frac{1}{2}n^2}_{\text{one LU fact}} + \underbrace{2n^2 n}_{n \text{ forw/backw.}}$

$$\sim \frac{8}{3}n^3 \qquad [\text{can be reduced to } 2n^3]$$

cost of applying $A^{-1}b_i$: $2n^2$ operations, is the same as for forw/backw substitution

overall cost for $A^{-1}b_i$ $i = 1, \dots k$:

$$\underbrace{2n^3}_{\text{inverse } A^{-1}} + \underbrace{2kn^2}_{\text{apply } A^{-1}}$$

always slower than using forward/backward substitution!