PROGRAM ROZWOJOWY
POLITECHNIKI WARSZAWSKIEJ

LECTURE 7

BASIC NUMERICAL METHODS FOR LINEAR ALGEBRAIC SYSTEMS

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Consider the system of n linear equations with n unknowns $x_1, x_2, ..., x_n$

$$\begin{cases} E_1: & a_{1,1}x_1 + a_{1,2}x_2 + ... + a_{1,n-1}x_{n-1} + a_{1,n}x_n = b_1 \\ E_2: & a_{2,1}x_1 + a_{2,2}x_2 + ... + a_{2,n-1}x_{n-1} + a_{2,n}x_n = b_2 \\ & \vdots \\ E_{n-1}: & a_{n-1,1}x_1 + a_{n-11,2}x_2 + ... + a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ E_n: & a_{n,1}x_1 + a_{n,2}x_2 + ... + a_{n,n-1}x_{n-1} + a_{n,n}x_n = b_n \end{cases}$$

In the matrix/vector notation, this system can be written as

$$\boldsymbol{Ax = b}$$

where $$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & ... & a_{2,n-1} & a_{2,n} \\ ... & ... & ... & ... & ... \\ a_{n-1,1} & a_{n-1,2} & ... & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & ... & a_{n,n-1} & a_{n,n} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

The Gauss Elimination is the systematic method to solve the linear system. The idea is simple: first the original system is transformed to the system with the **upper triangular matrix** (and the same solution), next this new system is solved directly by the **back-substitution procedure**.

Consider the first step of the transformation of the original system to the upper-triangular form. In this step, the unknown x_1 is eliminated from the equations E_2, E_3,..,E_n. To this aim, the equation E_1 is pre-multiplied by appropriate factors and subtracted from the other equations of the system. This transformation can be written as

$$\begin{cases} E_1: \ a_{1,1}x_1 + a_{1,2}x_2 + ... + a_{1,n}x_n = b_1 \\ E_2: \ a_{2,1}x_1 + a_{2,2}x_2 + ... + a_{2,n}x_n = b_2 \\ \vdots \\ E_n: \ a_{n,1}x_1 + a_{n,2}x_2 + ... + a_{n,n}x_n = b_n \end{cases} \Rightarrow \begin{cases} E_1: \ a_{1,1}x_1 + a_{1,2}x_2 + ... + a_{1,n}x_n = b_1 \\ E_2^{(1)}: \quad a_{2,2}^{(1)}x_2 + ... + a_{2,n}^{(1)}x_n = b_2^{(1)} \\ \vdots \\ E_n^{(1)}: \quad a_{n,2}^{(1)}x_2 + ... + a_{n,n}^{(1)}x_n = b_n^{(1)} \end{cases}$$

where

$$a_{i,j}^{(1)} = a_{i,j} - a_{1,j}l_{i,1} \quad , \quad l_{i,1} = \frac{a_{i,1}}{a_{1,1}}$$

$$b_i^{(1)} = b_i - b_1 l_{i,1} \quad , \quad i, j = 2,...,n$$

The matrix-vector form of the linear system obtained after the first step of the elimination process is $\boldsymbol{A}^{(1)}\boldsymbol{x} = \boldsymbol{b}^{(1)}$, where

$$\boldsymbol{A}^{(1)} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & \cdots & a_{2,n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} \end{bmatrix} \qquad \boldsymbol{b}^{(1)} = \begin{bmatrix} b_1 \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

This procedure is continued until the target form of the system is achieved. After $(k\text{-}1)^{th}$ step the system in the following form is obtained.

$$\begin{cases} E_1: & a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,k}x_k + \ldots + a_{1,n}x_n = b_1 \\ E_2^{(1)}: & \quad a_{2,2}^{(1)}x_2 + \ldots + a_{2,k}^{(1)}x_k + \ldots + a_{2,n}^{(1)}x_n = b_2^{(1)} \\ \vdots & \qquad \ddots \qquad \vdots \\ E_k^{(k-1)}: & \qquad\qquad a_{k,k}^{(k-1)}x_k + \ldots + a_{k,n}^{(k-1)}x_n = b_k^{(k-1)} \\ E_{k+1}^{(k-1)}: & \qquad\qquad a_{k+1,k}^{(k-1)}x_k + \ldots + a_{k+1,n}^{(k-1)}x_n = b_{k+1}^{(k-1)} \\ & \qquad\qquad \vdots \\ E_n^{(k-1)}: & \qquad\qquad a_{n,k}^{(k-1)}x_k + \ldots + a_{n,n}^{(k-1)}x_n = b_n^{(k-1)} \end{cases}$$

The upper index in the brackets indicates how many times each equation has been modified so far. In the k^{th} step, the equation $E_k^{(k-1)}$ is used to remove x_k from the equations $E_{k+1}^{(k-1)},\ldots,E_n^{(k-1)}$.

The corresponding transformation formulas are

$$a_{i,j}^{(k)} = a_{i,j}^{(k-1)} - a_{k,j}^{(k-1)} l_{i,k} \quad , \quad l_{i,k} = \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}}$$

$$b_i^{(k)} = b_i^{(k-1)} - b_k^{(k-1)} l_{i,k} \quad , \quad i, j = k+1,\ldots,n$$

and the matrix/vector form of the obtained system is $\boldsymbol{A}^{(k)}\boldsymbol{x} = \boldsymbol{b}^{(k)}$, where

$$\boldsymbol{A}^{(k)} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,k+1} & \cdots & a_{1,n} \\ * & \ddots & \cdots & \cdots & \vdots \\ * & * & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ * & * & \vdots & \ddots & \vdots \\ * & * & a_{n,k+1}^{(k)} & \vdots & a_{n,n}^{(k)} \end{bmatrix} \quad , \quad \boldsymbol{b}^{(k)} = [b_1, b_2^{(1)},..,b_k^{(k-1)},b_{k+1}^{(k)},..,b_n^{(k)}]^T$$

$*$ – *zero element*

The elimination stage of the solution procedure end after n-w step, when we finally arrived at the upper-triangular system

$$\begin{cases} E_1: & a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,k}x_k + \ldots + a_{1,n}x_n = b_1 \\ E_2^{(1)}: & \quad a_{2,2}^{(1)}x_2 + \ldots + a_{2,k}^{(1)}x_k + \ldots + a_{2,n}^{(1)}x_n = b_2^{(1)} \\ & \qquad \ddots \\ E_{n-1}^{(n-2)}: & \qquad\qquad a_{n-1,n-1}^{(n-2)}x_{n-1} + a_{n-1,n}^{(n-2)}x_n = b_{n-1}^{(n-2)} \\ E_n^{(n-1)}: & \qquad\qquad\qquad a_{n,n}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}$$

In the matrix/vector form, we have $\boldsymbol{U}\boldsymbol{x} = \hat{\boldsymbol{b}}$, where

$$\boldsymbol{U} \equiv \boldsymbol{A}^{(n-1)} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ * & a_{2,2}^{(1)} & \cdots & a_{2,n-1}^{(1)} & a_{2,n}^{(1)} \\ * & * & \ddots & \ddots & \vdots \\ * & * & * & a_{n-1,n-1}^{(n-2)} & a_{n-1,n}^{(n-2)} \\ * & * & * & * & a_{n,n}^{(n-1)} \end{bmatrix}, \quad \hat{\boldsymbol{b}} \equiv \boldsymbol{b}^{(n-1)} = [b_1, b_2^{(1)}, .., b_{n-1}^{(n-2)}, b_n^{(n-1)}]^T$$

The upper-triangular system can be solved directly by the **back-substitution procedure**. This algorithm consists in solving the equations **one by one** in the **reversed order**:

$$x_n = b_n^{(n-1)} / a_{n,n}^{(n-1)}$$

$$x_i = \frac{1}{a_{i,i}^{(i-1)}} \left[b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{i,j}^{(i-1)} x_j \right] \quad , \quad i = n-1, n-2, ..., 2, 1$$

The pseudo-code of the overall Gauss Elimination method has been presented in the next page.

Basic Gauss Elimination

```
% Stage 1 – Transformation to the triangular system
for k = 1 : n – 1
    for j = k + 1 : n
        L( j, k) ← A( j, k) / A(k, k)
        A( j, k + 1 : n) ← A( j, k + 1 : n) – L( j, k) · A(k, k + 1 : n)
        b( j) ← b( j) – L( j, k) · b(k)
    end
end
```

⇒

```
% Stage 2 – Solution
x(n) ← b(n) / A(n, n)
for j = n – 1 : 1 : –1
    x( j) ← b( j)
    for k = j + 1 : n
        x( j) ← x( j) – A( j, k) · x(k)
    end
    x( j) ← x( j) / A( j, j)
end
```

LU FACTORIZATION

We will show that the gauss Elimination method provides a very important "by-product" – the lower/upper triangular factorization of the matrix $\boldsymbol{A}$ (LU factorization).

Consider the first step of the elimination procedure. Using the numbers $l_{2,1}^{(1)}, l_{3,1}^{(1)}, \ldots, l_{n,1}^{(1)}$, one can construct the matrix $\tilde{\boldsymbol{L}}^{(1)}$

$$\tilde{\boldsymbol{L}}^{(1)} = \begin{bmatrix} 1 & * & * & * \\ -l_{2,1}^{(1)} & 1 & * & * \\ \vdots & * & \ddots & * \\ -l_{n,1}^{(1)} & * & * & 1 \end{bmatrix}, \quad * \text{ - zero element}$$

Note that the matrix of the system obtained after the first step of the elimination can be expressed as the following product $\boldsymbol{A}^{(1)} = \tilde{\boldsymbol{L}}^{(1)} \boldsymbol{A}$ (exercise for the Reader).

In the similar manner, the l-numbers defined during the k^{th} step of elimination can be used to define the matrix $\tilde{\boldsymbol{L}}^{(k)}$

$$\tilde{\boldsymbol{L}}^{(k)} = \begin{bmatrix} 1 & * & * & * & * & * \\ * & \ddots & * & * & * & * \\ * & * & 1 & * & * & * \\ * & * & -l_{k+1,k}^{(k)} & 1 & * & * \\ * & * & \vdots & * & \ddots & * \\ * & * & -l_{n,k}^{(k)} & * & * & 1 \end{bmatrix}$$

The matrix obtained in effect of this step can be expressed as $\boldsymbol{A}^{(k)} = \tilde{\boldsymbol{L}}^{(k)} \boldsymbol{A}^{(k-1)}$.

We immediately conclude the upper-triangular matrix **U** obtained in the end of the elimination procedure can be expressed by the formula

$$\boldsymbol{U} \equiv \boldsymbol{A}^{(n-1)} = \tilde{\boldsymbol{L}}^{(n-1)} \boldsymbol{A}^{(n-2)} = \underbrace{\tilde{\boldsymbol{L}}^{(n-1)} \tilde{\boldsymbol{L}}^{(n-2)} \ldots \tilde{\boldsymbol{L}}^{(2)} \tilde{\boldsymbol{L}}^{(1)}}_{\tilde{\boldsymbol{L}}} \boldsymbol{A}$$

An amazing fact is that **the matrix inverse to $\tilde{L}$ is lower triangular** and explicitly given as

$$\boldsymbol{L} \equiv \tilde{\boldsymbol{L}}^{-1} = \begin{bmatrix} 1 & * & * & * & * & * \\ l_{2,1}^{(1)} & 1 & * & * & * & * \\ l_{3,1}^{(1)} & l_{3,2}^{(2)} & \ddots & * & * & * \\ \vdots & \vdots & \ddots & 1 & * & * \\ \vdots & \vdots & \ddots & l_{n-1,n-2}^{(n-2)} & 1 & * \\ l_{n,1}^{(1)} & l_{n,2}^{(2)} & \cdots & l_{n,n-2}^{(n-2)} & l_{n,n-1}^{(n-1)} & 1 \end{bmatrix}$$

The proof of this fact can be found in the standard textbooks on numerical algebra, e.g., in the excellent book "Numerical Linear Algebra" by L.N. Trefethen and D. Bau.

Hence, we have obtained the following formula (***LU* factorization**)

$$\boldsymbol{A = LU}$$

The pseudo-code of the **basic version of the LU factorization** is presented in the next page.

Basic LU decomposition (*operates inside the original matrix*)

$$
\begin{cases}
\text{for } k = 1 : n-1 \\
\quad \text{for } j = k+1 : n \\
\qquad A(j,k) \leftarrow A(j,k) / A(k,k) \\
\qquad A(j,k+1:n) \leftarrow A(j,k+1:n) - A(j,k) \cdot A(k,k+1:n) \\
\quad \text{end} \\
\text{end}
\end{cases}
$$

Note :

1. *Original entries of* $\boldsymbol{A}$ *have been destroyed* !
2. *The main dianonal and the upper triangle of* $\boldsymbol{A}$ *contains nonzero elements of the factor* $\boldsymbol{U}$
3. *The lower triangle of* $\boldsymbol{A}$ *contains the elements of the factor* $\boldsymbol{L}$ (*diagonal elements of* $\boldsymbol{L}$ *are 1 and need not to be stored*)

Before we go further in our discussion, consider the numerical cost of the Gauss Elimination and *LU* factorization. Quick look at the summary of the GE algorithm leads to the following conclusions:

1. In the first stage (elimination) three nested loops have to be performed. Thus, roughly speaking, the number of floating point operations like multiplications and additions is proportional to the third power of the dimension of the system, i.e., proportional to $\boldsymbol{n^3}$. More precise result (assuming that n is sufficiently large) is that the number of the floating point operations (flops) is close to $\frac{2}{3}n^3$.

2. In the second stage (the back-substitution), two nested loops have to be performed. Thus, the number of the flops is proportional to $\boldsymbol{n^2}$. Hence, **for large values of *n* the numerical cost of the back-substitution is a negligible fraction of the numerical cost of the first stage (elimination)**.

Having in mind the estimation of the numerical cost, consider the situation when the sequence of the linear systems with the same matrix **A** but different right-hand side vectors has to be solved:

$$\boldsymbol{A}\boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)} \quad , \quad j = 1,..,m$$

If "ordinary" Gauss Elimination is used, the overall numerical cost of this tusk would be, roughly speaking, proportional to $m \cdot n^3$. The reason for this estimate is that any time a new system in the sequence is being solved the matrix **A** is transformed to the upper-triangular for at the cost proportional to n^3. However, this sequence of problems can be solved much more effectively using the **LU** factorization. The idea is to compute **L** and **U** factors only once. This way, the solution to each system in the sequence can be determined at the cost of two solutions of the triangular linear systems:

1) Compute factors $\boldsymbol{L}$ *and* $\boldsymbol{U}$ *of the matrix* $\boldsymbol{A}$ $(cost \sim n^3)$

2) for $j = 1,..,m$

$$\boldsymbol{L}\boldsymbol{U}\boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)} \Rightarrow \begin{cases} \boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}^{(j)} \\ \boldsymbol{U}\boldsymbol{x}^{(j)} = \boldsymbol{y} \end{cases}$$

The pseudo-code for this approach is shown in the next page.

Solution of the LU system

$\%\ \boldsymbol{Ly = b}$

$$
\begin{cases}
y(1) \leftarrow b(1) \\
\text{for } j = 2 : n \\
\quad y(j) \leftarrow b(j) \\
\quad \text{for } k = 1 : j - 1 \\
\qquad y(j) \leftarrow y(j) - A(j,k) \cdot y(k) \\
\quad \text{end} \\
\text{end}
\end{cases}
$$

$\Longrightarrow$

$\%\ \boldsymbol{Ux = y}$

$$
\begin{cases}
x(n) \leftarrow y(n) \,/\, A(n,n) \\
\text{for } j = n - 1 : 1 : -1 \\
\quad x(j) \leftarrow y(j) \\
\quad \text{for } k = j + 1 : n \\
\qquad x(j) \leftarrow x(j) - A(j,k) \cdot x(k) \\
\quad \text{end} \\
\quad x(j) \leftarrow x(j) \,/\, A(j,j) \\
\text{end}
\end{cases}
$$

THE CONCEPT OF PIVOTING

The basic variant of Gauss Elimination may fail even if the matrix A is nonsingular. Indeed, consider the k^{th} step of the elimination stage and assume that the unknown x_k has already disappeared from the equation $E_k^{(k-1)}$, i.e., the coefficient $a_{k,k}^{(k-1)}$ is equal zero. Then, none of the $l^{(k)}$ number cannot be calculated – we have division by zero!

Such situation occurs for instance (in the first and only elimination step) for any system with the matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

But this is not a whole story. Consider a small modification in the above example:

$$\boldsymbol{A} = \begin{bmatrix} 10^{-10} & 1 \\ 1 & 1 \end{bmatrix}$$

This time, the Gauss Elimination should work fine and the computations are hoped to yield the **LU** factors of **A**, i.e.:

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 \\ 10^{10} & 1 \end{bmatrix} \quad , \quad \boldsymbol{U} = \begin{bmatrix} 10^{-10} & 1 \\ 0 & 1-10^{10} \end{bmatrix}$$

Unfortunately, the computer's arithmetic is not exact. In our calculation, we will mimic real computations by assuming that the floating point numbers are represented in the exponential form (the base will be equal 10 instead of 2) and the **mantissa** is accurate up to the seventh digit after the decimal point.

On such "computer", the matrix element u_{22} will be evaluated as follows

$$10^{10} - 1 = 1 \cdot 10^{10} - 0.0000000001 \cdot 10^{10} = (1 - \underbrace{0.0000000}_{\textit{only these digits are resolved}} 001) \cdot 10^{10} \cong 10^{10}$$

Hence, the **exact U factor** will be in fact replaced by its approximation

$$\tilde{\boldsymbol{U}} = \begin{bmatrix} 10^{-10} & 1 \\ 0 & -10^{10} \end{bmatrix}$$

Note also that the factor **L** will be calculated exactly. Next, let us calculate the product of the factors **L** and $\tilde{\boldsymbol{U}}$:

$$\tilde{\boldsymbol{A}} = \boldsymbol{L}\tilde{\boldsymbol{U}} = \begin{bmatrix} 10^{-10} & 1 \\ 1 & 0 \end{bmatrix} \neq \boldsymbol{A}$$

As we see, the difference between $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ is significant!

Let us explore further consequences of this small difference between exact and computed $\boldsymbol{U}$ factors. Consider the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\,, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The exact solution is

$$\boldsymbol{x} = \begin{bmatrix} -(1-10^{-10})^{-1} \\ (1-10^{-10})^{-1} \end{bmatrix} \approx \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

If the (inexact) LU factorization is applied the computations will proceed as follows

$$\boldsymbol{L}\tilde{\boldsymbol{U}}\boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 10^{10} & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow \boldsymbol{y} = \begin{bmatrix} 1 \\ -10^{10} \end{bmatrix}$$

$$\begin{bmatrix} 10^{-10} & 1 \\ 0 & -10^{10} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -10^{10} \end{bmatrix} \Rightarrow \tilde{\boldsymbol{x}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \neq \boldsymbol{x}$$

!!!

Again, the significant discrepancy between exact and computed solution has been obtained. It also evident that the reason why our calculations fail to bring acceptable results is the **very large value of the $l_{2,1}^{(1)}$, which is equal 10^{10}!**

The remedy for this problem is (partial) **pivoting**, which is a "nick name" for the change of the equations' order. In our particular example, it works as follows

$$\boldsymbol{A}=\begin{bmatrix}10^{-10} & 1\\ 1 & 1\end{bmatrix},\ \boldsymbol{b}=\begin{bmatrix}1\\ 0\end{bmatrix} \quad \underset{\textit{pivoting}}{\Rightarrow} \quad \hat{\boldsymbol{A}}=\begin{bmatrix}1 & 1\\ 10^{-10} & 1\end{bmatrix},\ \hat{\boldsymbol{b}}=\begin{bmatrix}0\\ 1\end{bmatrix}$$

Now, the solution based on LU factorization works perfectly!

$$\hat{\boldsymbol{L}}=\begin{bmatrix}1 & 0\\ 10^{-10} & 1\end{bmatrix},\ \hat{\boldsymbol{U}}=\begin{bmatrix}1 & 1\\ 0 & 1-10^{-10}\end{bmatrix}\cong\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}=\tilde{\hat{\boldsymbol{U}}}$$

$$\hat{\boldsymbol{L}}\tilde{\hat{\boldsymbol{U}}}\boldsymbol{x}=\begin{bmatrix}0\\ 1\end{bmatrix} \Rightarrow \begin{bmatrix}1 & 0\\ 10^{-10} & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}0\\ 1\end{bmatrix} \Rightarrow \hat{\boldsymbol{y}}=\begin{bmatrix}0\\ 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}0\\ 1\end{bmatrix} \Rightarrow \tilde{\hat{\boldsymbol{x}}}=\begin{bmatrix}-1\\ 1\end{bmatrix}\approx\boldsymbol{x}$$

Generally speaking, in the k^{th} step of the elimination we search for the equation $E_m^{(k-1)}$, $m \in \{k, k+1, .., n\}$, such that the coefficients at x_k in this equation has the largest modulus. Then we interchange equations $E_m^{(k-1)}$ and $E_k^{(k-1)}$. This way, the **absolute values of all *l*-number do not exceed 1**. The catastrophic loss of accuracy will not appear.

Gauss Elimination with partial pivoting

$$
\begin{cases}
\textit{for } k = 1 : n-1 \\
\quad \textit{Select } i \geq k \textit{ such that } |A_{i,k}| \textit{ is maximal} \\
\quad A(k, k:n) \leftrightarrow A(i, k:n) \quad \textit{interchange rows } i \textit{ and } k \\
\quad b(k) \leftrightarrow b(i) \quad \textit{interchange elements of } \boldsymbol{b} \\
\quad \textit{for } j = k+1 : n \\
\qquad L(j,k) \leftarrow A(j,k) / A(k,k) \\
\qquad A(j, k+1:n) \leftarrow A(j, k+1:n) - L(j,k) \cdot A(k, k+1:n) \\
\qquad b(j) \leftarrow b(j) - L(j,k) \cdot b(k) \\
\quad \textit{end} \\
\textit{end}
\end{cases}
\Rightarrow
\begin{cases}
x(n) \leftarrow b(n) / A(n,n) \\
\textit{for } j = n-1 : 1 : -1 \\
\quad x(j) \leftarrow b(j) \\
\quad \textit{for } k = j+1 : n \\
\qquad x(j) \leftarrow x(j) - A(j,k) \cdot x(k) \\
\quad \textit{end} \\
\quad x(j) \leftarrow x(j) / A(j,j) \\
\textit{end}
\end{cases}
$$

What about the *LU* factorization when the pivoting is applied?

To understand what is happening, we need the concept of the permutation matrix. In general, the permutation matrix is obtained by changing the order of rows and/or columns of the identity matrix **I**. For example:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \boldsymbol{P} := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

What happens when a permutation matrix multiplies some other matrix **A**?. The following examples provide explanation:

$$\boldsymbol{PA} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 4 \end{bmatrix} - \textit{interchange of the rows 1 and 3}$$

$$\boldsymbol{AP} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix} - \textit{interchange of the collumns 1 and 3}$$

Consider now the **Gauss Elimination with pivoting**. Since the change of the equations' order happens before the *l*-numbers are determined, the transformation between intermediate matrixes $\boldsymbol{A}^{(k-1)}$ and $\boldsymbol{A}^{(k)}$ can be written in the following form

$$\boldsymbol{A}^{(0)} \equiv \boldsymbol{A} \;\;,\;\; \boldsymbol{A}^{(k)} = \tilde{\boldsymbol{L}}^{(k)} \boldsymbol{P}^{(k)} \boldsymbol{A}^{(k-1)} \;\;\;,\;\; k = 1,2,..,n-1$$

Then

$$\boldsymbol{U} \equiv \boldsymbol{A}^{(n-1)} = \tilde{\boldsymbol{L}}^{(n-1)} \boldsymbol{P}^{(n-2)} \boldsymbol{A}^{(n-2)} = \tilde{\boldsymbol{L}}^{(n-1)} \boldsymbol{P}^{(n-1)} \tilde{\boldsymbol{L}}^{(n-2)} \boldsymbol{P}^{(n-2)} ... \tilde{\boldsymbol{L}}^{(1)} \boldsymbol{P}^{(1)} \boldsymbol{A}$$

Another amazing property of the **P** and **L** factors is that the following equality holds

$$\tilde{\boldsymbol{L}}^{(n-1)} \boldsymbol{P}^{(n-1)} \tilde{\boldsymbol{L}}^{(n-2)} \boldsymbol{P}^{(n-2)} ... \tilde{\boldsymbol{L}}^{(1)} \boldsymbol{P}^{(1)} = \left[\tilde{\tilde{\boldsymbol{L}}}^{(n-1)} \tilde{\tilde{\boldsymbol{L}}}^{(n-2)} ... \tilde{\tilde{\boldsymbol{L}}}^{(1)} \right] \cdot \left[\boldsymbol{P}^{(n-1)} \boldsymbol{P}^{(n-2)} ... \boldsymbol{P}^{(1)} \right]$$

where

$$\tilde{\tilde{\boldsymbol{L}}}^{(k)} = \boldsymbol{P}^{(n-1)} ... \boldsymbol{P}^{(k+1)} \tilde{\boldsymbol{L}}^{(k)} [\boldsymbol{P}^{(k+1)}]^{-1} ... [\boldsymbol{P}^{(n-1)}]^{-1}$$

Moreover, the inverse matrix $\boldsymbol{L} = \left[\tilde{\tilde{\boldsymbol{L}}}^{(n-1)} \tilde{\tilde{\boldsymbol{L}}}^{(n-2)} ... \tilde{\tilde{\boldsymbol{L}}}^{(1)} \right]^{-1}$ is lower-triangular.

Introducing the overall permutation matrix $\boldsymbol{P} = \boldsymbol{P}^{(n-1)}\boldsymbol{P}^{(n-2)}\ldots\boldsymbol{P}^{(1)}$ we finally get the formula

$$\boldsymbol{PA} = \boldsymbol{LU}$$

The **LU factorization with pivoting** can be used to solve the linear system in the manner similar to what we have seen before

$$\boldsymbol{Ax} = \boldsymbol{b} \;\Rightarrow\; \boldsymbol{PAx} = \boldsymbol{Pb} \;\Rightarrow\; \boldsymbol{LUx} = \boldsymbol{Pb} \;\Rightarrow\; \begin{cases} \boldsymbol{Ly} = \boldsymbol{Pb} \\ \boldsymbol{Ux} = \boldsymbol{y} \end{cases}$$

Thus, the only complication is that we have to keep truck of the matrix rows reordering during elimination phase and then apply this reordering to the right-hand side vector (or vectors).

Other applications of the *LU* factorization include …

1. Evaluation of the matrix determinant (rarely needed – mostly academic problem)

$$\boldsymbol{PA} = \boldsymbol{LU} \quad \Rightarrow \quad \det(\boldsymbol{PA}) = \det(\boldsymbol{LU})$$

$$\Downarrow$$

$$\underbrace{\det(\boldsymbol{P})}_{\substack{1\,(\textit{even permut.}) \\ \textit{or } -1\,(\textit{odd perm.})}} \det(\boldsymbol{A}) = \underbrace{\det(\boldsymbol{L})}_{\substack{=1\,(\textit{product of} \\ \textit{diagonal elem.})}} \det(\boldsymbol{U})$$

$$\Downarrow$$

$$\det(\boldsymbol{A}) = \pm \det(\boldsymbol{U}) = u_{11} u_{22} \ldots u_{nn}$$

2. Explicit inversion of the matrix (needed very seldom, should be avoided in general)

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I} \quad \Rightarrow \quad \boldsymbol{A}\boldsymbol{x}_i = \boldsymbol{e}_i \;\; , \;\; \boldsymbol{e}_i \leftarrow i^{th} \; \textit{collumn of} \; \boldsymbol{I} \;\; , \;\; i = 1,2,..,n$$

$$\textit{Then} \;\; \boldsymbol{A}^{-1} = [\boldsymbol{x}_1 \,|\, \boldsymbol{x}_2 \,|\, \ldots \,|\, \boldsymbol{x}_n]$$

These systems are solved by the **LU** factorization at the overall cost proportional to n^3. Note that naive application of the Gauss Elimination for each system would give a total numerical cost proportional to n^4.

Final remarks:

- Other variants of the matrix factorizations exist. Some of them like **QR** factorization, Choleski factorization (exact and incomplete) are presented during the course "Numerical methods" (currently only in Polish).
- The Gauss Elimination and *LU* factorization methods belong to the class of – so called – exact methods for linear algebraic systems. „Exact" means noniterative, i.e. these methods (at least in theory) provide the exact solution in a finite number of well-defined steps. For large linear systems (say $n \sim 10^4$ or more) these methods are prohibitively expensive in both computer time and memory requirements). Fortunately, very large linear systems which arise in practical applications are most often sparse (each equation contains only a small number of unknowns) and they can be solved by specially design iterative methods. Several most popular iterative methods are presented during the course "Numerical methods".