

# Sisteme liniare - metode directe

Radu T. Trîmbițaș

16 martie 2025

## 1 Eliminare gaussiană

Să considerăm sistemul liniar cu  $n$  ecuații și  $n$  necunoscute

$$Ax = b, \quad (1)$$

unde  $A \in \mathbb{K}^{n \times n}$ ,  $b \in \mathbb{K}^{n \times 1}$  sunt date, iar  $x \in \mathbb{K}^{n \times 1}$  trebuie determinat, sau scris pe componente

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 & (E_1) \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 & (E_2) \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n & (E_n) \end{cases} \quad (2)$$

### 1.1 Eliminare gaussiană cu pivotare parțială

Metoda este dată de algoritmul 1.

### 1.2 Eliminare gaussiană cu pivot scalat pe coloană

O tehnică care micșorează eroarea și preîntâmpină anularea flotantă este *pivotarea parțială cu pivot scalat pe coloană*. Definim la început un factor de scară pentru fiecare linie

$$s_i = \max_{j=1, n} |a_{ij}| \text{ or } s_i = \sum_{j=1}^n |a_{ij}|.$$

---

**Algoritmul 1** Rezolvă sistemul  $Ax = b$  prin metoda eliminării a lui Gauss

---

**Intrare:** Matricea extinsă  $A = (a_{ij})$ ,  $i = \overline{1, n}$ ,  $j = \overline{1, n+1}$

**Ieșire:** Soluțiile  $x_1, \dots, x_n$  sau un mesaj de eroare

```
{Eliminare}
1: for  $i := 1$  to  $n - 1$  do
2:   Fie  $p$  cel mai mic întreg  $i \leq p \leq n$ ,  $a_{pi} \neq 0$  și  $|a_{pi}| = \max_{i \leq j \leq n} |a_{ji}|$ 
3:   if  $\nexists p$  then
4:     mesaj (' $\nexists$  soluție unică'); STOP
5:   end if
6:   if  $p \neq i$  then
7:      $(E_p) \leftrightarrow (E_i)$ 
8:   end if
9:   for  $j := i + 1$  to  $n$  do
10:     $m_{ji} := a_{ji}/a_{ii}$ ;
11:     $(E_j - m_{ji}E_i) \rightarrow (E_j)$ ;
12:  end for
13: end for
14: if  $a_{nn} = 0$  then
15:   mesaj (' $\nexists$  soluție unică'); STOP
16: end if
   {Substituție inversă}
17:  $x_n := a_{n,n+1}/a_{nn}$ ;
18: for  $i := n - 1$  downto  $1$  do
19:    $x_i = \left[ a_{i,n+1} - \sum_{j=i+1}^n a_{ij}x_j \right] / a_{ii}$ ;
20: end for
21: Returnează  $(x_1, \dots, x_n)$  {succes} STOP.
```

---

Dacă există un  $i$  a.î.  $s_i = 0$ , matricea este singulară. Pașii următori vor stabili interschimbările care se vor face. La al  $i$ -lea pas vom găsi cel mai mic întreg  $p$ ,  $i \leq p \leq n$ , a.î.

$$\frac{|a_{pi}|}{s_p} = \max_{i \leq j \leq n} \frac{|a_{ji}|}{s_j}$$

și apoi,  $(E_i) \leftrightarrow (E_p)$ . Scalarea ne garantează că cel mai mare element din fiecare coloană are înainte de comparațiile necesare pentru schimbare mărimea relativă 1. Scalarea se realizează doar în comparații, nu efectiv în matrice, astfel că împărțirea cu factorul de scalare nu produce nici o eroare de rotunjire.

## 2 Descompunere (factorizare) LUP

Ideea din spatele descompunerii LUP este de a găsi 3 matrice pătratice de ordinul  $n$   $L, U$  și  $P$  astfel încât

$$PA = LU \tag{3}$$

unde

- $L$  este o matrice triunghiulară inferior cu 1 pe diagonala principală;
- $U$  este o matrice triunghiulară superior;
- $P$  este o matrice de permutare.

Tripletul  $(L, U, P)$  se va numi **descompunere LUP** a matricei  $A$ . Orice matrice nesingulară posedă o astfel de descompunere.

Sistemul

$$Ax = b \tag{4}$$

se poate rezolva astfel

$$Ax = b \iff LUx = Pb \iff Ly = Pb \wedge Ux = y, \tag{5}$$

deoarece

$$Ax = P^{-1}LUx = P^{-1}Ly = P^{-1}Pb = b. \tag{6}$$

Având descompunerea LUP sistemul se poate rezolva cu algoritmul 2.

---

**Algoritmul 2** Rezolvă sistemul  $Ax = b$  având descompunerea LUP

---

**Intrare:** Matricele  $L$ ,  $U$ , vectorul  $b$ , vectorul de permutare  $\pi$ , toate de dimensiune  $n$

**Ieșire:** Soluțiile  $x_1, \dots, x_n$

```
1: for  $i := 1$  to  $n$  do
2:    $y_i := b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij}y_j$ ;
3: end for
4: for  $i := n$  downto  $1$  do
5:    $x_i = \left[ y_i - \sum_{j=i+1}^n u_{ij}x_j \right] / u_{ii}$ ;
6: end for
```

---

**Observație.** Am presupus că matricea  $P$  este reprezentată prin vectorul  $\pi$ .

Procedura care urmează (algoritmul 3) calculează descompunerea LUP. Ea reprezintă  $P$  ca un vector  $\pi$ , iar  $L$  și  $U$  sunt calculate în locul lui  $A$ , adică la terminare

$$a_{ij} = \begin{cases} l_{ij}, & \text{pentru } i > j, \\ u_{ij}, & \text{pentru } i \leq j. \end{cases}$$

---

**Algoritmul 3** Descompunere LUP

---

**Intrare:** Matricea  $A$ , de dimensiune  $m$

**Ieșire:** Matricele  $L$ ,  $U$  și  $P$ , toate de dimensiune  $m$

```
 $p = 1 : m$ ;
for  $k := 1$  to  $m - 1$  do
  {Pivotare}
  Alege  $i \geq k$  care maximizează  $|u_{ik}|$ ;
   $A_{k,:} \leftrightarrow A_{i,:}$ ; {interschimbare}
   $p_k \leftrightarrow p_i$ ;
   $lin := i + 1 : m$ ;
  {Calculez complementul Schur}
   $A_{lin,k} := A_{lin,k} / A_{k,k}$ ;
   $A_{lin,lin} := A_{lin,lin} - A_{lin,k}A_{k,lin}$ ;
end for
Extrage  $L$ ,  $U$ , generează  $P$ ;
```

---

**Exemplul 1.** Să se calculeze descompunerea LUP a matricei

```
A =  
 2.0000  0      2.0000  0.6000  
 3.0000  3.0000  4.0000 -2.0000  
 5.0000  5.0000  4.0000  2.0000  
-1.0000 -2.0000  3.4000 -1.0000
```

Calculule decurg astfel

```
>> [l,u,p]=lup(A)  
interschimb liniile 1 și 3
```

```
A =  
  
 5.0000  5.0000  4.0000  2.0000  
 3.0000  3.0000  4.0000 -2.0000  
 2.0000  0      2.0000  0.6000  
-1.0000 -2.0000  3.4000 -1.0000
```

calculez complementul Schur

```
A =  
 5.0000  5.0000  4.0000  2.0000  
 0.6000  3.0000  4.0000 -2.0000  
 0.4000  0      2.0000  0.6000  
-0.2000 -2.0000  3.4000 -1.0000
```

```
A =  
 5.0000  5.0000  4.0000  2.0000  
 0.6000  0      1.6000 -3.2000  
 0.4000 -2.0000  0.4000 -0.2000  
-0.2000 -1.0000  4.2000 -0.6000
```

interschimb liniile 2 și 3

```
A =  
 5.0000  5.0000  4.0000  2.0000  
 0.4000 -2.0000  0.4000 -0.2000  
 0.6000  0      1.6000 -3.2000  
-0.2000 -1.0000  4.2000 -0.6000
```

calculez complementul Schur

$$A = \begin{pmatrix} 5.0000 & 5.0000 & 4.0000 & 2.0000 \\ 0.4000 & -2.0000 & 0.4000 & -0.2000 \\ 0.6000 & 0 & 1.6000 & -3.2000 \\ -0.2000 & 0.5000 & 4.2000 & -0.6000 \end{pmatrix}$$

$$A = \begin{pmatrix} 5.0000 & 5.0000 & 4.0000 & 2.0000 \\ 0.4000 & -2.0000 & 0.4000 & -0.2000 \\ 0.6000 & 0 & 1.6000 & -3.2000 \\ -0.2000 & 0.5000 & 4.0000 & -0.5000 \end{pmatrix}$$

interschimb liniile 3 și 4

$$A = \begin{pmatrix} 5.0000 & 5.0000 & 4.0000 & 2.0000 \\ 0.4000 & -2.0000 & 0.4000 & -0.2000 \\ -0.2000 & 0.5000 & 4.0000 & -0.5000 \\ 0.6000 & 0 & 1.6000 & -3.2000 \end{pmatrix}$$

calculez complementul Schur

$$A = \begin{pmatrix} 5.0000 & 5.0000 & 4.0000 & 2.0000 \\ 0.4000 & -2.0000 & 0.4000 & -0.2000 \\ -0.2000 & 0.5000 & 4.0000 & -0.5000 \\ 0.6000 & 0 & 0.4000 & -3.2000 \end{pmatrix}$$

$$A = \begin{pmatrix} 5.0000 & 5.0000 & 4.0000 & 2.0000 \\ 0.4000 & -2.0000 & 0.4000 & -0.2000 \\ -0.2000 & 0.5000 & 4.0000 & -0.5000 \\ 0.6000 & 0 & 0.4000 & -3.0000 \end{pmatrix}$$

Rezultatele finale sunt

```

l =
    1.0000  0      0      0
    0.4000  1.0000  0      0
   -0.2000  0.5000  1.0000  0
    0.6000  0      0.4000  1.0000

u =
    5.0000    5.0000    4.0000    2.0000
     0      -2.0000    0.4000   -0.2000
     0       0      4.0000   -0.5000
     0       0       0      -3.0000

p =
    0  0  1  0
    1  0  0  0
    0  0  0  1
    0  1  0  0

```

Verificare:

```

>> disp(l*u)

    5.0000    5.0000    4.0000    2.0000
    2.0000    0.0000    2.0000    0.6000
   -1.0000   -2.0000    3.4000   -1.0000
    3.0000    3.0000    4.0000   -2.0000

>> disp(p*A)

    5.0000    5.0000    4.0000    2.0000
    2.0000    0.0000    2.0000    0.6000
   -1.0000   -2.0000    3.4000   -1.0000
    3.0000    3.0000    4.0000   -2.0000

```

### 3 Descompunere (factorizare) Cholesky

O matrice hermitiană și pozitiv definită se poate factoriza sub forma  $A = LL^*$  sau  $A = R^*R$ , unde  $L$  este o matrice triunghiulară inferior, iar  $R$  este triunghiulară superior.

Pentru algoritm a se vedea notele de curs sau algoritmul 4.

---

**Algoritmul 4** Descompunere Cholesky

---

**Intrare:** Matricea  $A$ , hermitiană și pozitiv definită

**Ieșire:** Matricea  $R$ , triunghiulară superior

```
 $R := A;$   
for  $k := 1$  to  $m$  do  
  for  $j := k + 1$  to  $m$  do  
     $R_{j,j:m} := R_{j,j:m} - R_{k,j:m} \overline{R_{k,j}} / R_{k,k}$   
  end for  
   $R_{k,k:m} := R_{k,k:m} / \sqrt{R_{k,k}}$   
end for
```

---

### 4 Probleme

**Problema 1.** Implementați eliminarea gaussiană cu pivotare parțială sau scalată pe coloană (la alegere) în MATLAB.

**Problema 2.** Să se implementeze descompunerea LUP. Să se scrie rutine pentru rezolvarea unui sistem folosind descompunerea LUP.

**Problema 3.** Generați sisteme cu matrice aleatoare nesingulare ce au soluția  $[1, \dots, 1]^T$ . Rezolvați-le cu eliminare gaussiană și descompunere LUP.

**Problema 4.** Să se scrie rutine pentru descompunerea Cholesky a unei matrice hermitiene și pozitiv definite și rezolvarea unui sistem cu o astfel de matrice prin descompunere Cholesky. Testați rutinele pentru matrice generate aleator și sisteme cu matrice aleatoare, dar cu soluție cunoscută.

**Problema 5.** Rezolvați sistemul

$$\begin{bmatrix} 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \\ 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -19 \\ -34 \\ 16 \\ 26 \end{bmatrix}.$$



**Problema 6.** Calculați descompunerile LU și Cholesky ale matricei

$$\begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix}.$$

**Problema 7.** Rezolvați sistemul

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 1 \\ -1 & 1 & 0 & \dots & 1 \\ -1 & -1 & 1 & \dots & 1 \\ -1 & -1 & -1 & 1 & \dots & 1 \\ \vdots & & \ddots & \ddots & \vdots \\ -1 & \dots & -1 & -1 & 1 \end{bmatrix} x = \begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \\ \vdots \\ -n + 2 \end{bmatrix}$$

prin descompunere LUP și QR. Ce se observă? Explicați.

## 5 Probleme suplimentare

**Problema 8.** Scrieți rutine pentru descompunerea LUP în care permutarea să se facă fizic și logic (cu vectori de permutări) și comparați timpii de execuție al ambelor variante pentru sisteme cu dimensiunea între 100 și 300.

**Problema 9.** Implementați rezolvarea unui sistem liniar prin descompunere QR.