

FACTORIZAREA MATRICILOR

Factorizarea reprezintă operația de trecere de la o mulțime la mulțimea claselor de echivalentă, în raport cu o relație de echivalentă dată. Factorizarea se aplică, la rezolvarea sistemelor de ecuații liniare. În esență, factorizarea este o procedură de transformare a unei matrici, în general pătratice, într-o matrice superior triunghiulară (elementele superioare diagonalei principale sunt diferite de zero, în timp ce, elementele inferioare acesteia sunt toate egale cu zero) sau una inferior triunghiulară (elementele inferioare diagonalei principale sunt diferite de zero, în timp ce, elementele superioare acesteia sunt toate egale cu zero). Metodele directe pentru rezolvarea sistemelor (de forma $\mathbf{Ax}=\mathbf{B}$), cele mai utilizate, sunt metodele bazate pe procesul de eliminare sau descompunere a matricei coeficienților (matricea \mathbf{A}). Astfel, pentru un sistem $\mathbf{Ax}=\mathbf{B}$:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases},$$

în care coeficienții determină definirea matricei $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$,

matricea coeficienților; $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ este vectorul necunoscutelor; $\mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ este

vectorul termenilor liberi ai sistemului, metoda generală de transformare într-un sistem echivalent, "Metoda de eliminare a lui Gauss", se prezintă în continuare.

Prima transformare aplicată sistemului $\mathbf{Ax}=\mathbf{B}$, are ca efect eliminarea necunoscutei x_1 din ultimele $n-1$ ecuații, adunând prima ecuație la următoarele $n-1$, înmulțită cu $\left(\frac{-a_{11}^{(1)}}{a_{11}^{(1)}}\right)$, obținându-se un nou sistem de ecuații: $\mathbf{A}^{(1)}\mathbf{x}=\mathbf{B}^{(1)}$. În a doua transformare, se folosește a doua ecuație, din sistemul $\mathbf{A}^{(1)}\mathbf{x}=\mathbf{B}^{(1)}$, înmulțită

cu $\left(\frac{-a_{12}^{(2)}}{a_{22}^{(2)}} \right)$, pentru eliminarea necunoscutei x_2 din ultimele **n-2** ecuații ale acestui

sistem, obținându-se un nou sistem: $A^{(2)}x = B^{(2)}$. Se observă că înainte de eliminarea necunoscutei x_k , sistemul echivalent, obținut după eliminarea necunoscutelor x_1, x_2, \dots, x_{k-1} , poate fi scris sub forma:

$$A^{(k)}x = B^{(k)}, \quad k=1,2, \dots, n$$

a cărui descriere analitică este:

$$A^{(k)} = \begin{pmatrix} a_{i,j}^{(k)} \end{pmatrix}, \quad B^{(k)} = \begin{bmatrix} b_1^{(k)} \\ b_2^{(k)} \\ \vdots \\ b_n^{(k)} \end{bmatrix},$$

iar elementele matricei $A^{(k)}$ sunt:

$$a_{i,j}^{(k)} = \begin{cases} a_{i,j}^{(k-1)}, & i \leq k-1 \\ 0, & k, j \leq k-1 \\ a_{i,j}^{(k-1)} - \frac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} a_{k-1,j}^{(k-1)}, & i \geq k, j \geq k \end{cases},$$

respectiv, elementele vectorului $B^{(k)}$, se calculează cu relațiile:

$$b_i^{(k)} = \begin{cases} b_i^{(k-1)}, & i \leq k-1 \\ b_i^{(k-1)} - \frac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} b_{k-1}^{(k-1)}, & i \geq k \end{cases}$$

Pentru $k=n$ transformări, sistemul nou obținut este:

$$\left\{ \begin{array}{l} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \dots + a_{1k}^{(1)}x_k + \dots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ a_{21}^{(2)}x_1 + a_{22}^{(2)}x_2 + \dots + a_{2k}^{(2)}x_k + \dots + a_{2n}^{(2)}x_n = b_2^{(2)} \\ \vdots \\ a_{kk}^{(k)}x_k + \dots + a_{kn}^{(k)}x_n = b_k^{(k)} \\ \vdots \\ a_{nn}^{(n)}x_n = b_n^{(n)} \end{array} \right.$$

De remarcat, în scrierea acestui sistem, în care s-a realizat eliminarea succesivă a coeficienților ecuațiilor sistemului, că, această operație nu se poate realiza dacă un coefficient, din linia "p" și coloana "p" se anulează $a_{pp}^{(p)} = 0$. Pentru a se evita această situație există *strategii de pivotare*, care nu modifică rezultatele calculelor și soluțiilor sistemelor.

Pivotarea, pentru evitarea situației în care coeficientul din linia "p" se anulează ($a_{pp}^{(p)} = 0$), acesta fiind, din acest motiv neutilizabil în metoda eliminării, a lui Gauss, și apoi la rezolvarea sistemului prin metoda substituției, este necesară, și se realizează prin căutarea acelui coefficient, din linia "k", $k > p$, pentru care $a_{kp}^{(p)} \neq 0$, apoi realizându-se inter-schimbarea acelor linii ("p" și "k"), fiind posibilă, în acest mod, alegerea acestui coefficient, drept pivot, în metoda eliminării. Pivotarea se realizează, uneori, și pentru minimizarea erorilor de calcul. Pentru a ilustra aceste observații se va rezolva un sistem simplu, de două ecuații și două necunoscute, în care se va pivota linia a doua.

Pentru a evidenția importanța pivotării liniilor sau coloanelor, în procesul de rezolvare a sistemelor de ecuații, se propune rezolvarea unui sistem de două ecuații cu două necunoscute, în care coeficienții sunt scrise cu trei cifre semnificative după virgulă: $\begin{cases} 1,133 \cdot x_1 + 5,281 \cdot x_2 = 6,414 \\ 24,14 \cdot x_1 - 1,210 \cdot x_2 = 22,93 \end{cases}$. Pentru aceasta se elimină coefficientul necunoscutei x_1 din cea de-a două ecuație, prin adunarea la cea de-a două ecuație a primei ecuații înmulțită cu numărul $-\frac{24,14}{1,133} = -21,31$, a cărei valoare se aproximează la două cifre semnificative. Se obține rezultatul:

$$\begin{cases} 1,133 \cdot x_1 + 5,281 \cdot x_2 = 6,414 \\ 113,7 \cdot x_2 = 113,8 \end{cases}, \text{ cu soluțiile } x_2 = 1,001, \text{ respectiv } x_1 = 0,9956.$$

Dacă se schimbă liniile acestui sistem, între ele, și se rezolvă sistemul, prin eliminarea aceluiși coefficient, din linia a două (fostă linia întâi), considerând cinci cifre semnificative după virgulă ale factorului de multiplicare,

$$-\frac{1,133}{24,14} = -0,04693, \quad \text{atunci se obține sistemul redus:}$$

$$\begin{cases} 24,14 \cdot x_1 - 1,210 \cdot x_2 = 22,93 \\ 5,338 \cdot x_2 = 5,338 \end{cases}, \text{ cu soluțiile } x_2=1,000, \text{ respectiv } x_1=1,000. \text{ Există,}$$

prin urmare, diferențe între soluțiile acestui sistem, chiar dacă nu s-a modificat, decât în mică măsură, doar la nivelul reținerii numărului de cifre semnificative ale factorului de multiplicare, în procesul de eliminare a coeficienților.

Strategia de pivotare constă în realizarea unui număr de etape obligatorii necesare translației linilor unei matrice pătratice, astfel încât, respectând anumite convenții și reguli de calcul, să se determine factorii de multiplicare și reducere la zero a elementelor matricei date pentru descompunerea fără propagarea erorilor de calcul, sau cu erori minime. Astfel, se impune verificarea magnitudinii valorii absolute a elementelor fiecărei coloane, aflate pe, sau, sub diagonala principală a matricei. Se localizează poziția elementului care verifică această condiție:

$$|a_{kp}| = \max\{|a_{pp}|, |a_{p+1p}|, \dots, |a_{N-1p}|, |a_{Np}|\}$$

și apoi se schimbă pozițiile liniei "p" cu linia "k", dacă $k > p$. Se împarte coloana respectivă cu valoarea maximă găsită, rezultând valori, mai mici sau egale cu valoarea 1, în valoare absolută. Alegerea, ca pivot, a celui mai mare dintre elementele coloanelor unei matrice, implică propagarea unor erori mici sau chiar efectuarea calculelor fără erori.

Pentru rezolvarea unui sistem de n ecuații cu n necunoscute, prin metoda eliminării și substituției, a lui Gauss, (în forma generală), se poate folosi următorul fișier funcție (**substitutie.m**), în MATLAB:

```
function X=substitutie(A,B)
```

% Argumente de intrare

% A=matricea (patraticea) superior triunghiulară a coeficientilor

% B=matricea coloana a termenilor liberi

% Argumente de ieșire

% X=matricea soluțiilor sistemului

```

A=input('Introduceti matricea patraticea triunghiulara a
coeficientilor:');
B=input('Introduceti matricea coloana a termenilor liberi:');
n=length(B);
X=zeros(n,1);
X(n)=B(n)/A(n,n);
for k=n-1:-1:1
if A(k,k)~=0
    X(k)=(B(k)-A(k,k+1:n)*X(k+1:n))/A(k,k);
else
    disp('Atentie! Impartire cu zero!')
end
end
X

```

Această funcție este construită astfel încât să rezolve un sistem de **n** ecuații cu **n** necunoscute în cazul în care matricea coeficienților sistemului admite triunghiularizarea, și această matrice, poate fi introdusă, la solicitarea programului în forma triunghiularizată superior. În caz contrar, adică, dacă matricea coeficienților se introduce ne-triunghiularizată, programul va afișa soluțiile determinate prin simpla împărțire a matricilor sistemului, fără a se realiza verificările necesare rezolvării corecte a sistemului. Fișierul nu realizează triunghiularizarea matricei pătratice a coeficienților, din acest motiv, un fișier funcție, **triun_sup.m**, este necesar pentru realizarea acestei operațiuni simple (triunghiularizarea superioară a matricei coeficienților), și acesta ar putea fi:

```

function X=triun_sup(A,B)

% Argumente de intrare
% -A=matricea patraticea a coeficienților
% -B=matricea coloana a termenilor liberi

A=input('Introduceti matricea patraticea a coeficienților:');
B=input('Introduceti matricea coloana a termenilor liberi');

% Argumente de ieșire
% -X=matricea coloana a soluțiilor sistemului
[N,N]=size(A);

```

```
X=zeros(N,1);
C=zeros(1,N+1);
```

% Se defineste matricea extinsa prin adaugarea matricei coeficientilor

```
A_ext=[A B];
for p=1:N-1
```

% Pivotarea parțială a coloanei "p"

```
[Y,j]=max(abs(A_ext(p:N,p)));
```

% Interschimbarea liniilor "p" și "j"

```
C=A_ext(p,:);
A_ext(p,:)=A_ext(j+p-1,:);
A_ext(j+p-1,:)=C;
if A_ext(p,p)==0
```

'Matricea "A" este singulară. Nu există soluție unică!'

```
break
```

```
end
```

% Eliminarea coloanei "p"

```
for k=p+1:N
```

```
    m=A_ext(k,p)/A_ext(p,p);
```

```
    A_ext(k,p:N+1)=A_ext(k,p:N+1)-m*A_ext(p,p:N+1);
end
```

```
end
```

'MATRICEA SUPERIOR TRIUNGHIALARA EXTINSA ESTE:'

A_ext

'MATRICEA SUPERIOR TRIUNGHIALARA ESTE:'

A=A_ext(1:N,1:N)

'MATRICEA COEFICIENTILOR ESTE:'

B=A_ext(1:N,N+1)

% Aplicarea metodei substitutiei pentru rezolvarea sistemului

'SOLUTIA SISTEMULUI ESTE'

```
X=substitutie(A_ext(1:N,1:N),A_ext(1:N,N+1));
```

De remarcat că ultima linie a fișierului funcție **triun_sup.m** realizează apelarea fișierului **substitutie.m**, pentru rezolvarea sistemului de **n** ecuații și **n** necunoscute, în cazul în care se poate triunghiulariza superior matricea coeficienților. În fișierul **substitutie.m**, se va realiza o modificare a sintaxei acestuia, pentru a se evita solicitarea utilizatorului în vederea introducerii matricilor sistemului, în sensul că liniile care realizează acest lucru vor fi anulate sau vor fi marcate cu caracterul "%". În acest mod MATLAB nu va mai executa acele instrucțiuni, fișierul **substitutie.m** va avea următoarea sintaxă (modificarea acestui fișier se va realiza "a priori"):

```
function X=substitutie(A,B)
```

% Argumente de intrare

*%-A=matricea (patratica) superior triunghiulara a coeficientilor
%-B=matricea coloana a termenilor liberi*

% Argumente de ieșire

%-X=matricea soluțiilor sistemului

```
%A=input('Introduceti matricea patratica triunghiulara a coeficientilor:');
%B=input('Introduceti matricea coloana a termenilor liberi:');
```

```
n=length(B);
X=zeros(n,1);
X(n)=B(n)/A(n,n);
for k=n-1:-1:1
if A(k,k)~=0
    X(k)=(B(k)-A(k,k+1:n)*X(k+1:n))/A(k,k);
else
    disp('Atentie! Impartire cu zero!')
end
end
X
```

Pentru exemplificare se va rezolva sistemul $\begin{cases} 2 \cdot x_1 + 4 \cdot x_2 - 6 \cdot x_3 = -4 \\ x_1 + 5 \cdot x_2 + 3 \cdot x_3 = 10, \\ x_1 + 3 \cdot x_2 + 2 \cdot x_3 = 5 \end{cases}$

folosind cele două fișiere:

» triun_sup

Introduceti matricea patratica a coeficientilor:[2,4,6;1,5,3;1,3,2]

Introduceti matricea coloana a termenilor liberi:[-4;10;5]

MATRICEA SUPERIOR TRIUNGHIULARA EXTINSA ESTE:

A_ext =

$$\begin{matrix} 2 & 4 & 6 & -4 \\ 0 & 3 & 0 & 12 \\ 0 & 0 & -1 & 3 \end{matrix}$$

MATRICEA SUPERIOR TRIUNGHIULARA EXTINSA ESTE:

A =

$$\begin{matrix} 2 & 4 & 6 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{matrix}$$

MATRICEA COEFICIENTILOR ESTE:

B =

$$\begin{matrix} -4 \\ 12 \\ 3 \end{matrix}$$

SOLUTIA SISTEMULUI ESTE

X =

$$\begin{matrix} -1 \\ 4 \\ -3 \end{matrix}$$

Se va aplica același program pentru rezolvarea sistemului

$$\left\{ \begin{array}{l} x_1 + x_2 = 5 \\ 2 \cdot x_1 - x_2 + 5 \cdot x_3 = -9 \\ 3 \cdot x_2 - 4 \cdot x_3 + 2 \cdot x_4 = 19 \\ 2 \cdot x_3 + 6 \cdot x_4 = 2 \end{array} \right. , \text{obtinându-se rezultatele:}$$

» triun_sup

Introduceti matricea patratica a coeficientilor:

[1 1 0 0; 2 -1 5 0; 0 3 -4 2; 0 0 2 6]

Introduceti matricea coloana a termenilor liberi:[5; -9; 19; 2]

MATRICEA SUPERIOR TRIUNGHIULARA EXTINSA ESTE:

$A_{ext} =$

2.0000	-1.0000	5.0000	0	-9.0000
0	3.0000	-4.0000	2.0000	19.0000
0	0	2.0000	6.0000	2.0000
0	0	0	0.5000	0.5000

MATRICEA SUPERIOR TRIUNGHIULARA ESTE:

$A =$

2.0000	-1.0000	5.0000	0
0	3.0000	-4.0000	2.0000
0	0	2.0000	6.0000
0	0	0	0.5000

MATRICEA COEFICIENTILOR ESTE:

$B =$

-9.0000
19.0000
2.0000
0.5000

SOLUTIA SISTEMULUI ESTE

$X =$

2
3
-2
1

Revenind la procesul de propagare a erorilor de calcul, datorat factorizării matricilor, cu sau fără pivotarea liniilor, se propune aici un fișier funcție, **fact_pivot.m**, care realizează factorizarea unei matrici pătratice, cu pivotarea liniilor acesteia. Fișierul este astfel construit încât să rezolve sistemul de **n** ecuații cu **n** necunoscute, folosind fișierul funcție **substitutie.m**. Pentru execuția fișierului

este necesară introducerea, de la tastatură, a matricei coeficienților, respectiv a matricii termenilor liberi, pentru un sistem de $n \times n$. Fișierul este:

```

function X=fact_pivot(A,B);

% Argumente de intrare
% A=matricea patratica a coeficientilor: size(A)=n
% B=matricea coloana a termenilor liberi: size(B)=(n, 1)

A=input('Introduceti matricea patratica a coeficientilor:');
B=input('Introduceti matricea coloana a termenilor liberi:');

% Argumente de iesire
% X=matricea coloana a solutiilor sistemului
% A=matricea pivotata compusa din matricile L si U : A=LU

%NOTATII
% L= matricea inferior triunghiulara
% U= matricea superior triunghiulara
% Y= este matricea coloana care verifica relatia LY=B
% C= matricea temporara folosita pentru calcule si pivotari
% R= matricea pivotarilor

% Se initializeaza matricile X si Y, matricea temporara C si matricea R
% care contine informatii privind numarul de pivotari

[N,N]=size(A);
X=zeros(N,1);
Y=zeros(N,1);
C=zeros(1,N);
R=1:N;

for p=1:N-1
% Se cauta elementul MAXIM al coloanei "p"

[max1,j]=max(abs(A(p:N,p)));
% Se interschimba liniile "p" si "j"

```

```
C=A(p,:);
A(p,:)=A(j+p-1,:);
A(j+p-1,:)=C;
d=R(p);
R(p)=R(j+p-1);
R(j+p-1)=d;

if A(p,p)==0
    'Matricea "A" este singulara. Nu exista solutie unica!'
    break
end
```

% Se calculeaza factorul de multiplicare pentru termenii aflati sub
% diagonala principala a matricei A

```
for k=p+1:N
    factor=A(k,p)/A(p,p);
    A(k,p)=factor;
    A(k,p+1:N)=A(k,p+1:N)-factor*A(p,p+1:N);
end
end
```

% Se rezolva sistemul pentru gasirea solutiei Y=inv(L)*B
% L este matricea inferior- triunghiulara obtinuta din A(k,p+1:N)

```
Y(1)=B(R(1));
for k=2:N
    Y(k)=B(R(k))-A(k,1:k-1)*Y(1:k-1);
```

% Rezolvarea sistemului in X folosind rutina **substitutie.m**

```
X=substitutie(A,Y);
end
disp('.....')
disp(' R E Z U L T A T E L E   S U N T : ')
disp('.....')
disp(' ')
disp(' ')
```

```

disp('MATRICEA PIVOTATA (A=LU) :')
A
disp(' ')
disp('MATRICEA COLOANA A TRANSFORMARII LY=B :')
Y
disp(' ')
disp('MATRICEA SOLUTIILOR :')
X
disp(['.....'])

```

De exemplu, pentru rezolvarea sistemului

$$\begin{cases} x_1 + x_2 = 5 \\ 2 \cdot x_1 - x_2 + 5 \cdot x_3 = -9 \\ 3 \cdot x_2 - 4 \cdot x_3 + 2 \cdot x_4 = 19 \\ 2 \cdot x_3 + 6 \cdot x_4 = 2 \end{cases}$$
, se folosește succesiunea:

» fact_pivot

Introduceti matricea patratica a coeficientilor:

[1,1,0,0;2,-1,5,0;0,3,-4,2;0,0,2,6]

Introduceti matricea coloana a termenilor liberi:[5;-9;19;2]

|.....|
| R E Z U L T A T E L E S U N T : |
|.....|

MATRICEA PIVOTATA (A=LU) :

A =

2	-1	5	0
0	3	-4	2
0	0	2	6
1/2	1/2	-1/4	1/2

MATRICEA COLOANA A TRANSFORMARII LY=B :

Y =
-9
19
2
2
1/2

MATRICEA SOLUȚIILOR :

X =
2
3
-2
1
|.....|

Dacă o matrice $A \in \mathbb{R}^{n \times n}$ poate fi scrisă sub formă unui produs de două matrici, LU , unde matricea L este inferior triunghiulară iar matricea U este superior triunghiulară, în condițiile în care determinanții parțiali (minorii) matricei, sunt nenuli, $\det[a_{11}] \neq 0$, $\det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \neq 0$, ..., $\det[A] \neq 0$, atunci, se spune că

aceea matrice se poate factoriza, prin descompunere după algoritmi verificăți. Descompunerea este unică dacă elementele matricei L sau U , de pe diagonala principală sunt specificate astfel:

$$A = L \cdot U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ \vdots & & & & \\ l_{n1} & l_{n2} & \dots & l_{n,n-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & & & & \\ 0 & 0 & 0 & 0 & u_{nn} \end{bmatrix}$$

Dacă se realizează produsul dintre cele două matrici L și U , și se identifică termenii, se obțin n^2 ecuații neliniare cu n^2 necunoscute. Se observă că :

$$u_{11} = a_{11}, u_{12} = a_{12}, \dots, u_{1n} = a_{1n}$$

iar, dacă se determină elementele primei coloane de sub diagonala principală, se obțin ecuațiile:

$$l_{21}s_{11} = a_{21}; l_{31}s_{11} = a_{21}, \dots, l_{n1}s_{11} = a_{n1}]$$

Forma generală a ecuațiilor din care rezultă elementele liniei k , din matricea U , este:

$$\left. \begin{array}{l} \sum_{j=1}^{k-1} l_{kj}u_{j,k} + u_{k,k+1} = a_{kk} \\ \sum_{j=1}^{k-1} l_{kj}u_{j,k+1} + u_{k,k+1} = a_{k,k+1} \\ \vdots \\ \sum_{j=1}^{k-1} l_{kj}u_{jn} + u_{kn} = a_{kn} \end{array} \right\} \quad \begin{array}{l} u_{k,k} = a_{k,k} - \sum_{j=1}^{k-1} l_{kj}u_{jk} \\ u_{k,k+1} = a_{k,k+1} - \sum_{j=1}^{k-1} l_{kj}u_{j,k+1} \\ \vdots \\ u_{kn} = a_{kn} - \sum_{j=1}^{k-1} l_{kj}u_{jn} \end{array}$$

Elementele coloanei k a matricei L (inferioare diagonalei principale), sunt date de ecuațiile:

$$\left. \begin{array}{l} \sum_{j=1}^{k-1} l_{k+1,j}u_{j,k} + l_{k+1,k}u_{k,k} = a_{k+1,k} \\ \sum_{j=1}^{k-1} l_{k+2,j}u_{j,k} + l_{k+2,k}u_{k,k} = a_{k+2,k} \\ \vdots \\ \sum_{j=1}^{k-1} l_{n,j}u_{j,k} + l_{n,k}u_{k,k} = a_{n,k} \end{array} \right\} \quad \begin{array}{l} l_{k+1,k} = \frac{1}{u_{k,k}} \left(a_{k+1,k} - \sum_{j=1}^{k-1} l_{k+1,j}u_{jk} \right) \\ l_{k+2,k} = \frac{1}{u_{k,k}} \left(a_{k+2,k} - \sum_{j=1}^{k-1} l_{k+2,j}u_{jk} \right) \\ \vdots \\ l_{n,k} = \frac{1}{u_{k,k}} \left(a_{n,k} - \sum_{j=1}^{k-1} l_{n,j}u_{jk} \right) \end{array}$$

MATLAB folosește trei metode de bază, pentru factorizarea matricilor:

- *Factorizarea Cholesky*-pentru matrici simetrice, pozitiv definite

- **Factorizarea Lower Upper (LU)-eliminarea Gauss**, pentru matrici pătratice
- **Factorizarea ortogonală (QR)**- pentru matrici rectangulare

Acste metode de factorizare se apelează, în MATLAB, utilizând funcțiile specifice: **chol()**, **lu()**, **qr()**. Caracteristica, valabilă în toate metodele, o constituie utilizarea matricei triunghiulare ale carei elemente, situate deasupra sau sub diagonala principală, sunt toate egale cu zero. Toate sistemele de ecuații liniare, care permit utilizarea matricilor triunghiulare (factorizări), se rezolvă ușor folosind *pre-* sau *post-substituția*.

FACTORIZAREA CHOLESKY

Factorizarea Cholesky este o metodă utilizată pentru rezolvarea unui sistem de ecuații liniare de forma: $\mathbf{S}\mathbf{X}=\mathbf{B}$, unde \mathbf{S} este o *matrice pozitiv definită* (o matrice este pozitiv definită dacă elementele diagonalei principale sunt toate pozitive, iar celelalte elemente componente ale matricei nu sunt “prea mari”). Reprezentarea schematică, a factorizării Cholesky, este prezentată în figura alăturată, cu ajutorul blocurilor de calcul DSP (Digital Signal Processing), proiectate, special pentru procesarea semnalelor digitale, pentru a fi folosite în mediile de simulare dinamică în *SIMULINK* (MATLAB 6)

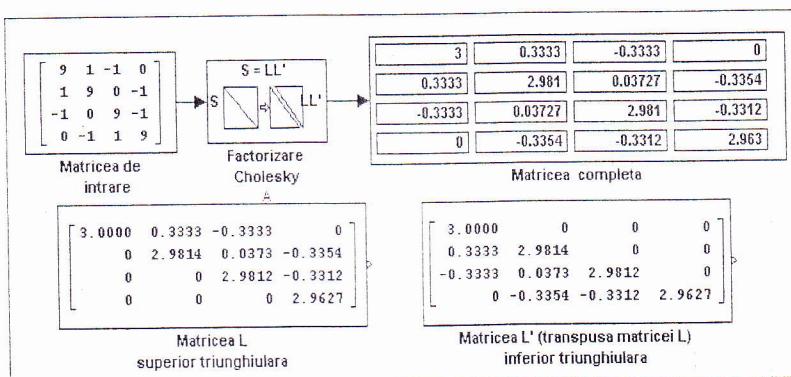


Fig. 4.1 – Schema bloc Simulink- factorizarea Cholesky

pozitiv definită. În acest caz soluția dată prin Solverul Cholesky este eronată, încăcă matricea coeficienților nu este pozitiv definită. Totuși, rădăcinile verifică prima ecuație, cu eroare la a patra zecimală. Rezolvarea sistemului prin determinarea matricei inverse (matricea coeficienților fiind nesingulară, iar rangul matricei extinse este egal cu rangul matricei coeficienților, aşadar sistemul admite soluție unică), are ca rezultat vectorul soluție: $X=[-5; \quad 2; \quad 6; \quad -4;]$.

*Notă:-Dacă se încearcă determinarea factorului Cholesky, prin funcția MATLAB **chol([1,2,1,1;2,1,3,3;1,1,1,1;2,7,2,3])**, va fi afișat un mesaj de eroare:*

(??? Error using ==> chol
Matrix must be positive definite.)

încăcă matricea de la argumentul funcției **chol()** are elemente cu “mult diferite”.

FACTORIZAREA LOWER-UPPER

Prin factorizarea **lower-upper**, o matrice pătratică este descompusă sub forma produsului a două matrici triunghiulare: una inferior triunghiulară (lower), permuată (o matrice permuată, după o anumită regulă este o matrice în care liniile sunt pivotate între ele), cu elementele, de pe diagonala principală, egale cu unitatea (matricea L) și cealaltă, o matrice superior triunghiulară (upper), (matricea U). În MATLAB, factorizarea **lu** este utilizată pentru obținerea inversei unei matrici cu funcția **inv()** (inversa unei matrici A, se calculează ca produsul dintre inversa matricei inferior triunghiulară L, calculată prin eliminare Gauss, și inversa matricei superior triunghiulară U, calculată prin aceeași metodă), respectiv pentru calculul determinantului matricelor cu funcția **det()** (ca produs dintre determinantul matricei inferior triunghiulară L și determinantul matricei superior triunghiulară U). Este, totodată, baza rezolvării ecuațiilor liniare prin "împărțirea matricelor" obținută cu operatorii „\”, și „/”.

Permutarea liniilor sau coloanelor unei matrici se poate exemplifica foarte sugestiv printr-o schemă cu blocuri DSP. Presupunem că se dorește permutarea

liniilor unei matrici $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$ după următorul index: $P=[2,1,3,3,2,2,]$,

respectiv permutarea coloanelor, după următoarea regulă: $Q=[2,1,3,2,2]$. Atunci, schema bloc este următoarea:

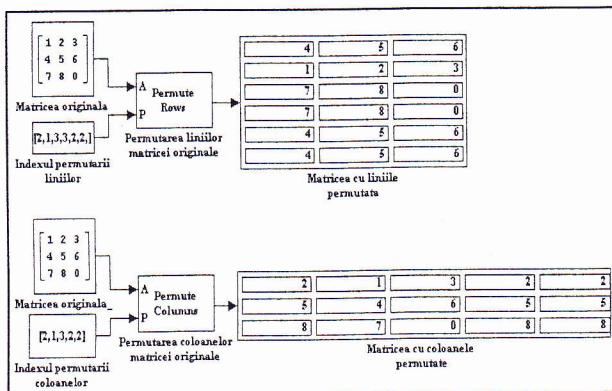


Fig. 4.5 – Permutarea matricilor

Factorizarea **lower-upper** se realizează după următoarea schemă generalizată:

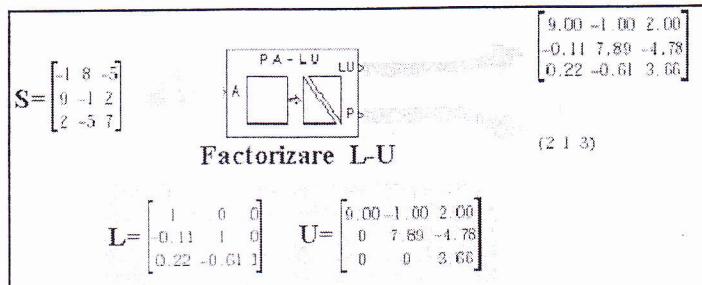


Fig. 4.6 – Schema generală de factorizare **lower- upper**

Factorizarea unei matrici, prin metoda **lower-upper**, se realizează cu funcția **lu()** și se apelează cu una dintre sintaxele:

1. $[LU]=lu(S)$ - cu un singur parametru de ieșire, matricea $[LU]$
2. $[L,U]=lu(S)$ – returnează o matrice superior triunghiulară, U , și o matrice inferior triunghiulară permuatată, L , astfel încât $S=L*U$
3. $[L,U,P]=lu(X)$ – returnează o matrice superior triunghiulară, U , respectiv matricea inferior triunghiulară, L , și permutarea matriceală, P , astfel încât $L*P=P*S$

Ex. : Să se factorizeze, prin metoda **lower-upper**, matricea $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$

```
» A=[1,2,3;4,5,6;7,8,0];
» [LU]=lu(A)
LU =
    7.0000   8.0000      0
    0.1429   0.8571   3.0000
    0.5714   0.5000   4.5000
```

```
» [L,U]=lu(A)
L =
    0.1429   1.0000      0
    0.5714   0.5000   1.0000
    1.0000       0      0
```

```
U =
    7.0000   8.0000      0
        0   0.8571   3.0000
        0       0   4.5000
```

unde, matricea L, este o permutare a matricei inferior triunghiulare, iar U este o matrice superior triunghiulară.

```
» [L,U,P]=lu(A)
L =
    1.0000       0      0
    0.1429   1.0000      0
    0.5714   0.5000   1.0000
U =
    7.0000   8.0000      0
        0   0.8571   3.0000
        0       0   4.5000
```

```
P =
    0   0   1
    1   0   0
    0   1   0
```

În Matlab, versiunile superioare versiunii 5.3, este posibilă factorizarea matricilor pătratice, prin metoda **lower-upper**, folosind pachetul de programe **DSP Blockset** din **Simulink**.

De exemplu, pentru matricea $A = \begin{bmatrix} 1 & -2 & 3 \\ 4 & 0 & 6 \\ 2 & -1 & 3 \end{bmatrix}$, folosind

pachetul **DSP Blockset**, se obține matricea compusă din matricea **L**, inferior triunghiulară, și matricea superior triunghiulară:

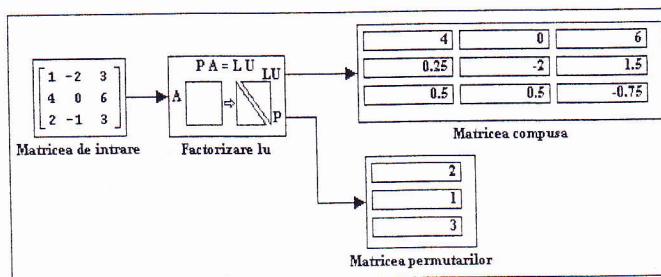


Fig. 4.7 – Factorizarea și permutarea matricilor cu DSP

Matricea A_p , (linia 2 pivotată, astfel încât aceasta devine linia 1) este

$$A_p = \begin{bmatrix} 4 & 0 & 6 \\ 1 & -2 & 3 \\ 2 & -1 & 3 \end{bmatrix}, \text{ iar matricile triunghiulare } L \text{ (nepivotată) respectiv matricea } U,$$

$$\text{superior triunghiulară, sunt: } L = \begin{bmatrix} 1 & 0 & 0 \\ 0.25 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix}, \text{ respectiv } U = \begin{bmatrix} 4 & 0 & 6 \\ 0 & -2 & 1.5 \\ 0 & 0 & -0.75 \end{bmatrix}.$$

Pentru a observa modul în care sunt determinate matricile **L** și **U**, se va apela funcția MATLAB **lu(A)** în toate cele trei variante posibile:

```

» A=[1,-2,3;4,0,6;2,-1,3];
» [LU]=lu(A)
LU =
    4.0000    0   6.0000
    0.2500  -2.0000  1.5000
    0.5000   0.5000 -0.7500

```

» [L,U]=lu(A)

L =

$$\begin{matrix} 0.2500 & 1.0000 & 0 \\ 1.0000 & 0 & 0 \\ 0.5000 & 0.5000 & 1.0000 \end{matrix}$$

U =

$$\begin{matrix} 4.0000 & 0 & 6.0000 \\ 0 & -2.0000 & 1.5000 \\ 0 & 0 & -0.7500 \end{matrix}$$

» [L,U,P]=lu(A)

L =

$$\begin{matrix} 1.0000 & 0 & 0 \\ 0.2500 & 1.0000 & 0 \\ 0.5000 & 0.5000 & 1.0000 \end{matrix}$$

U =

$$\begin{matrix} 4.0000 & 0 & 6.0000 \\ 0 & -2.0000 & 1.5000 \\ 0 & 0 & -0.7500 \end{matrix}$$

P =

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Rezolvarea sistemelor de ecuații $AX=B$ prin factorizarea lower upper presupune următoarele etape:

i. determinarea matricelor L și U cu funcția MATLAB

»[L,U]=LU(A);

ii. determinarea vectorilor »Y=L\B și »X=U\Y ;

Etapa ii., poate fi realizată și compact cu instruțiunea:

» X=U\ (L\B)

De exemplu, pentru sistemul de 3 ecuații și 3 necunoscute

$$\begin{cases} x - 2y + 3z = 1 \\ 4x + 6z = -2 \\ 2x - y + 3z = -1 \end{cases}$$

```
»A=[1,-2,3;4,0,6;2,-1,3];B=[1;-2;-1];
»[L,U]=lu(A);X=U\ (L\B)
```

obținându-se soluția sistemului :

X =

```
-2
0
1
```

Dacă factorizarea se realizează prin apelarea cu trei parametri de ieșire, atunci se obțin *rădăcinile permutate* ale sistemului dat :

```
»[L,U,P]=lu(C); X=U\ (L\B)
```

X =

```
-0.5000
1.5000
0.5000
```

În variantele superioare versiunii MATLAB 5.3 este posibilă rezolvarea sistemelor liniare cu **n** ecuații și **n** necunoscute prin metoda **lower-upper** folosind pachetul de programe **DSP Blockset** din **Simulink**. De exemplu, pentru rezolvarea

sistemului 3×3 $\begin{cases} x - 2y + 3z = 1 \\ 4x + 6z = -2 \\ 2x - y + 3z = -1 \end{cases}$, se folosește următorul model **Simulink** :

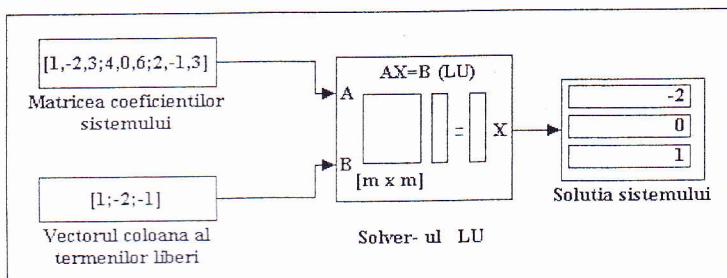


Fig. 4.8 – Rezolvarea sistemelor folosind solverul *lower- upper*