MEF-AI.1.2

REZOLVAREA SISTEMELOR DE ECUAŢII LINIARE

CUPRINS

MEF-AI.1.2.1 INTRODUCERE

Fundamentarea teoretică riguroasă din punct de vedere matematic, a Metodei Elementelor Finite (MEF), presupune cunoştinţe avansate de modelare matematică analitică şi numerică, în corelaţie. Deoarece această lucrare nu se adresează celor implicaţi în cercetarea teoretică fundamentală a MEF, în acest capitol se prezintă elemente de rezolvarea sistemelor de ecuaţii liniare necesare înţelegerii, implementării şi aplicării acestei metode de către utilizatori şi programatori.

Sistemul de n ecuaţii liniare, cu n necunoscute,

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

se poate scrie sintetic sub formele:

$$[A]\,[X] = [B]\ ,\ [X]\,[A] = [B] \quad 2)$$

în care, $[A] = [a_{ij}]_{n,n}$ este matricea sistemului; $[X] = [x_i]_{n,1}$ – matricea coloană a necunoscutelor şi $[B] = [b_i]_{n,1}$ – matricea coloană a termenilor liberi.

Determinarea soluţiei sistemului (1) se poate realiza *direct*, fără inversarea matricei sistemului prin transformări echivalente în vederea găsirii unor sisteme particulare (diagonale, triunghiulare) pentru care soluţiile se găsesc uşor sau *indirect*, cu relaţia

$$[X] = [A]^{-1}\,[B]. \quad (3)$$

Pe de altă parte, metodele de rezolvare a sistemelor liniare pot fi: *exacte*, când soluţia se determină într-un număr finit şi bine determinat de operaţii aritmetice sau *aproximative (iterative)*, când soluţia se determină într-un număr nedefinit de operaţii matematice, în funcţie de o precizia dorită.

Metodele exacte (Gauss, Gauss-Jordan etc.), care implică multe operaţii aritmetice, acumulări mari ale erorilor de rotunjire, timp mare de calculator şi necesităţi de memorie mari, fac ca în final, precizia teoretică scontată să nu se realizeze [Vraciu, 1982] fiind folosite la rezolvarea sistemelor de ordin nu prea mare ($n < 1000$).

Metodele iterative (Jacobi, Gauss-Seidel etc.) sunt mai rapide decât cele exacte, necesită un număr mai mic de operaţii aritmetice, putând fi aplicate cu succes şi la rezolvarea sistemelor de dimensiuni mari ($n > 50$) când metodele exacte devin costisitoare. Aceste metode fiind mult mai eficiente, in primul rând, ca viteză de calcul. s-au impus, odată cu creşterea memoriei centrale (RAM) a calculatoarelor.

MEF-AI.1.2.2 METODA DE ELIMINARE (TRUNGHIULARIZARE) A LUI GAUSS

Metoda lui Gauss este una din cele mai vechi şi larg răspândită metodă directă şi exactă de rezolvare a sistemelor cu ecuaţii liniare. Dintre metodele exacte este metoda care necesită cele mai puţine operaţii matematice, ceea ce explică utilizarea ei frecvent în programele de element finit, în diferite variante. Numărul mai mic de operaţii conduce la un timp de calcul scăzut şi la precizie mărită, deoarece erorile din timpul rezolvării se acumulează de la o operaţie aritmetică la alta [Munteanu, 1981]. Ideea de bază a acestei metode constă în aducerea prin transformări elementare a matricei sistemului la forma echivalentă, superior triunghiulară şi apoi prin substituirea succesivă, în sens invers, determinarea necunoscutelor.

Triunghiularizarea matricei sistemului se va realiza folosind algoritmul prezentat în cap. MEF-AI.1.1). Modificarea succesivă corespunzătoare a elementelor matricei termenilor liberi se va realiza cu relaţia

$$b_i^{(k)} = b_i^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} b_k^{(k-1)}, \tag{4}$$

în care k = 1,2,…, n-1 şi i = k+1, k+2,…, n.

Astfel, după realizarea pasului k = n-1, sistemul (1) devine

$$\begin{bmatrix} a_{11} & a_{12} & . & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \dots & \dots & & \dots & \dots \\ 0 & 0\dots\dots & & \dots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(3)} \\ . \\ b_n^{(n-1)} \end{bmatrix}. \tag{5}$$

Pornind de la ultima ecuaţie prin înlocuiri succesive (retrosubstituţia), rezultă

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}},$$

$$x_{n-1} = \frac{b_{n-1}^{(n-2)} - a_{n-1,n}^{(n-2)} x_n}{a_{n-1,n-1}^{(n-2)}},$$

……

$$x_{n-k} = \frac{b_{n-k}^{(n-k-1)} - \sum_{l=n-k+1}^{n} a_{n-k,l}^{(n-k-1)} x_l}{a_{n-k,n-k}^{(n-k-1)}}, \tag{6}$$

……

$$x_1 = \frac{b_1 - \sum_{l=2}^{n} a_{1l} x_l}{a_{11}},$$

cu k = 1, 2,…, n-1.

Dacă matricea sistemului este simetrică, $a_{ij} = a_{ji}$, relaţiile de triunghiularizare (4) devin:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ki}^{(k-1)} a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}},$$

$$b_i^{(k)} = b_i^{(k-1)} - \frac{a_{ki}^{(k-1)}}{a_{kk}^{(k-1)}} b_k^{(k-1)}, \tag{7}$$

în care k = 1, 2,…, n-1; i = k+1, k+1,…n; j = k+1, k+2,…, n. Acest algoritm lucrează numai cu coeficienţii de deasupra diagonalei principale (inclusiv ai acesteia), ceea ce implică o mare economie de memorie şi micşorare a timpului de calcul.

În fig. 1 se prezintă, subprogramele GAUSS şi GAUSS_S care au la bază metoda eliminării a lui Gauss pentru rezolvarea sistemelor liniare cu matrice nesingulară şi, respectiv, în plus simetrică.

Dacă coeficienţii şi necunoscutele sunt numere întregi, rezultatul obţinut prin această metodă este exact. De cele mai multe ori se lucrează cu numere zecimale. Calculatorul lucrează însă cu un număr limitat de zecimale după virgulă, 7 sau 14 zecimale pentru cazul calculelor în simplă precizie sau respectiv dublă precizie. Prin urmare, se introduce o eroare de rotunjire, care la un sistem cu multe ecuaţii nu poate fi neglijată. În continuare, se prezintă o *metodă de ameliorare a preciziei de calcul*, de asemenea prin metoda eliminării a lui Gauss.

Se consideră că datorită erorilor de rotunjire menţionate mai sus, în urma rezolvării sistemului de ecuaţii prin metoda eliminării, între soluţia aproximativă obţinută $\left[\overline{X}\right]$ şi soluţia exactă există relaţia,

$$[X]=\left[\overline{X}\right]+[\delta], \tag{8}$$

în care [δ] reprezintă matricea abaterilor (corecţiilor) datorită erorilor de rotunjire. Înlocuind relaţia (8) în (2), rezultă reziduul (eroarea) soluţiei aproximative

$$[\varepsilon]=[A][\delta], \tag{9}$$

dat şi de relaţia,

$$[\varepsilon]=[B]-[A]\left[\overline{X}\right]. \tag{10}$$

Algoritmul determinării abaterii [δ] a soluţiei obţinute $\left[\overline{X}\right]$ constă în rezolvarea sistemului (9) – de asemenea cu metoda Gauss – în care reziduul [ε] s-a calculat cu relaţia (10).

Metoda eliminării a lui Gauss permite găsirea soluţiei unui sistem de ecuaţii liniare de ordinul n, într-un număr finit de paşi, necesitând un număr de operaţii elementare de ordinul n^3. Odată cu creşterea ordinului sistemului, erorile de rotunjire se pot acumula dramatic, ducând la erori relative mari ale soluţiei. Pentru minimizarea erorilor, pivotul (elementul a_{kk} de pe diagonala principală a matricei sistemului) trebuie să fie cât mai mare, deci, în consecinţă, se impune reordonarea ecuaţiilor sistemului după fiecare etapă. Acest procedeu de pivotare cu reordonarea implică un număr important de operaţii suplimentare în cazul sistemelor mari.

MEF-AI.1.2.3 METODA DESCOMPUNERII A LUI CHOLESKY

Deoarece, *în cazul MEF*, adeseori, matricea sistemului de ecuaţii este simetrică, cu elementele de pe diagonala principală pozitive (matricea este pozitiv definită), se poate aplica metoda Cholesky de triunghiularizare a matricei sistemului, prezentată în cap. MEF-AI.1.1.

Sistemul de ecuaţii liniare (2), ţinând cont de relaţia (25, cap. MEF-AI.1.1), devine

$$[S]^T[S][X]=[B] \tag{11}$$

şi se poate descompune în două sisteme de ecuaţii liniare, cu matricele triunghiulare:

$$[S]^T[U]=[B], \tag{12}$$

$$[S][X]=[U]. \tag{13}$$

Rezolvând sistemul (12) prin substituţie, rezultă elementele matricei coloană [U] şi ţinând cont de aceasta, prin retrosubstituţie, din (13) se determină matricea necunoscută [X]. Substituţia şi retrosubstituţia se efectuează la fel ca în cazul metodei eliminării a lui Gauss.

```
Procedure GAUSS(var n:integer;a:mat;r:vect);
{type mat=array[1..n,1..n] of real;vect=array[1..n] of real;}
var i,j,l,il:integer;
begin
   {triunghiularizare}
   for i:=1 to n-1 do
      begin
         if a[i,i] <> 0 then
            begin
               l:=i+1;
               r[i]:=r[i]/a[i,i];
               for j:=l to n do a[i,j]:=a[i,j]/a[i,i];
               for il:=l to n do
                  begin
                     r[il]:=r[il]-a[il,i]*r[i];
                     for j:=l to n do
                        a[il,j]:=a[il,j]-a[il,i]*a[i,j];
                  end;
            end;
      end;
   r[n]:=r[n]/a[n,n];
   {retrosubstituţia}
   for i:=n-1 downto 1 do
      for j:=i+1 to n do r[i]:=r[i]-a[i,j]*r[j];
end;
```

```
Procedure GAUSS_S(var n,nm:integer;a:vect1;r:vect2);
{matricea simetrică  a sistemului de ordinul n se organizează  sub
forma unui vector de ordinul nm = n*(n+1)/2, care conţine numai
elementele părţii superioare}
{type vect1=array[1..nm] of real;vect2=array[1..n] of real;}
var nn,n1,mm,m1,mi,im,ij,mj,i,j,m,nx,il,j1,nt:integer;
{triunghiularizarea}
begin
   nn:=n;
   n1:=n-1;
   mm:=1;
   for m:=1 to n do
      begin
         m1:=m+1;
         if a[m] <> 0 then
            begin
               r[m]:=r[m]/a[mm];
               if m < n then
                  begin
                     mi:=mm;
                     for i:=m1 to n do
                        begin
                           mi:=mi+1;
                           a[mi]:=a[mi]/a[mm];
                        end;
                     im:=mm;
                     ij:=mi;
                     for i:=m1 to n do
                        begin
                           im:=im+1;
                           r[i]:=r[i]-a[im]*a[mm]/r[m];
                           mj:=im;
                           for j:=i to n do
                              begin
                                 ij:=ij+1;
                                 a[ij]:=a[ij]-a[im]*a[mm]/a[mj];
                                 mj:=mj+1;
                              end;
                        end;
                  end;
               if m < n then
                  begin
                     mm:=mm+nn;
                     nn:=nn-1;
                  end;
            end;
      end;
   nt:=Trunc(n*(n+1)/2);
   nx:=2;
   for i:=1 to n1 do
      begin
         nt:=nt-nx;
         ij:=nt;
         il:=n-i;
         j1:=il;
         for j:=1 to i do
            begin
               ij:=ij+1;
               j1:=j1+1;
               r[il]:=r[il]-a[ij]*r[j1];
            end;
         nx:=nx+1;
      end;
end;
```

Fig. 1

În lucrarea [Munteanu, 1982] se prezintă detaliat subprogramul de rezolvare a unui sistem de ecuaţii liniare, care are matricea cu proprietăţile menţionate mai sus.

Metoda descompunerii a lui Cholesky, ca şi metoda triunghiularizării a lui Gauss, este o metodă directă, exactă de rezolvare a sistemelor de ecuaţii liniare.

MEF-AI.1.2.4 METODA APROXIMAŢIILOR SUCCESIVE (JACOBI)

Această metodă permite găsirea soluţiei unui sistem de ecuaţii liniare, pornind de la o *aproximaţie iniţială a soluţiei*, pe baza unui proces iterativ. Restricţiile de utilizare a acestei metode sunt legate de asigurarea convergenţei procesului iterativ. *Convergenţa* (apropierea soluţiei aproximative de soluţia exactă) este asigurată dacă elementele de pe diagonala principală a matricei sistemului sunt nenule şi mai mari în modul decât modulul celorlalte elemente ale liniei respective (matrice dominant diagonală).

Presupunând elementele diagonalei principale nenule, rezolvând prima ecuaţie a sistemului (1) în raport cu x_1, a doua în raport cu x_2 şi aşa mai departe, a n-a în raport cu x_n, se obţine sistemul echivalent

$$[X]=[\beta]+[\alpha][X], \tag{14}$$

în care,

$$[\alpha]=\begin{bmatrix}\alpha_{11} & \alpha_{12} & \dots & \alpha_{1n}\\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n}\\ \dots & \dots & \dots & \dots\\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nn}\end{bmatrix}, [\beta]=\begin{bmatrix}\beta_1\\ \beta_2\\ .\\ \beta_n\end{bmatrix}, \tag{15}$$

cu

$$\alpha_{ij}=-\frac{a_{ij}}{a_{ii}},\ \beta_i=\frac{b_i}{a_{ii}}, \tag{16}$$

pentru: i, j = 1,2,…, n; $i \neq j$; $\alpha_{ii} = 0$; a_i, b_i – elementele matricelor sistemului (2).

Procesul aproximaţiilor succesive porneşte cu soluţia aproximativă iniţială (de ordinul zero)

$$\left[X^{(0)}\right]=[\beta], \tag{17}$$

care înlocuită în (1.14) generează soluţia aproximativă de ordinul unu,

$$\left[X^{(1)}\right]=[\beta]+[\alpha]\left[X^{(0)}\right]. \tag{18}$$

Continuând înlocuirile succesive, se obţine soluţia aproximativă de ordinul k,

$$\left[X^{(k)}\right]=[\beta]+[\alpha]\left[X^{(k-1)}\right] \tag{19}$$

şi se poate considera că soluţia exactă a sistemului este

$$[X]=\lim_{k\to\infty}\left[X^{(k)}\right]. \tag{20}$$

Sintetic, relaţiile procesului iterativ se scriu sub forma:

$$\begin{aligned} x_i^{(0)} &= \beta_i,\\ x_i^{(k)} &= \beta_i + \sum_{j=1, i\neq 1}^{n} \alpha_{ij} x_j^{(k-1)}, \end{aligned} \tag{21}$$

Unde, i = 1, 2,…, n şi k = 1, 2,….

Procesul iterativ descris de aceste relaţii se continuă până când eroarea absolută

$$\Delta_i^{(k)} = x_i^{(k)} - x_i^{(k-1)} \tag{22}$$

respectă relaţia

$$\max\left|\Delta_i^{(k)}\right| \le \varepsilon, \tag{23}$$

în care, ε reprezintă eroarea maximă admisibilă, prescrisă.

Implementarea sub formă de program a algoritmului Jacobi este greoaie şi, în continuare, se prezintă metoda Gauss-Seidel, care are şi avantaje legate de convergenţă şi necesarul de memorie.

MEF-AI.1.2.5 METODA GAUSS-SEIDEL

Această metodă este o modificare a metodei Jacobi, în scopul realizării avantajelor prezentate mai sus, care constă în calculul componentei aproximative a soluţiei sistemului, de la pasul k, $x_i^{(k)}$ în funcţie de componentele $x_1^{(k)}, x_2^{(k)}, ..., x_{i-1}^{(k)}$, deja calculate în cadrul acestui pas şi componentele $x_i^{(k-1)}, x_{i+1}^{(k-1)}, ..., x_n^{(k-1)}$, de la pasul anterior.

Astfel, soluţiile aproximărilor succesive la pasul k = 1, 2, … se pot calcula cu relaţiile:

$$\begin{aligned}
x_1^{(k)} &= \beta_1 + \sum_{j=1}^{n} \alpha_{1j} x_j^{(k-1)}, \\
x_2^{(k)} &= \beta_2 + \alpha_{21} x_1^{(k)} + \sum_{j=2}^{n} \alpha_{2j} x_j^{(k-1)}, \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_i^{(k)} &= \beta_i + \sum_{j=1}^{i-1} \alpha_{ij} x_j^{(k)} + \sum_{j=1}^{n} \alpha_{ij} x_j^{(k-1)}, \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n^{(k)} &= \beta_n + \sum_{j=1}^{n-1} \alpha_{nj} x_j^{(k)} + \alpha_{nn} x_n^{(k-1)}.
\end{aligned} \tag{24}$$

Procesul iterativ descris de aceste relaţii şi în cazul acestei metode continuă până când se verifică relaţia (23).

Pentru realizarea unei convergenţe sigure, indiferent de alegerea soluţiei iniţiale, este necesară transformarea sistemului iniţial într-un sistem care are matricea simetrică şi pozitiv definită. Aceasta se obţine înmulţind la stânga, cu $[A]^T$, ecuaţia matriceală a sistemului liniar (2), care are matricea $[A]$ nesingulară [Beu, 1992]. Astfel, sistemul de rezolvat devine,

$$[A]^T [A][X] = [A]^T [B]. \tag{25}$$

În fig. 2 se prezintă, subprogramele SEIDEL şi SEIDEL_T destinate rezolvării sistemelor liniare, fără transformarea, şi, respectiv, cu transformarea matricei sistemului în vederea asigurării convergenţei.

MEF-AI.1.2.6 REZOLVAREA SISTEMELOR DE ECUAŢII LINIARE CU RESTRICŢII

Procedeele matematice de rezolvare a unui sistem de ecuaţii cu restricţii sunt multiple. Cele mai utilizate

metode sunt *eliminarea* unui număr de ecuaţii egal cu numărul condiţiilor de restricţie, metoda *multiplicatorilor Lagrange* şi metoda *funcţiei de penalizare.*

```
Procedure SEIDEL(var n:integer;a:mat;b:vect;er:real);
{type mat=array[1..n,1..n] of real;vect=array[1..n] of real;}
var i,j,k,kmax:integer;
    beta:vect;s,emax:real;
begin
   kmax:=100;
   for i:=1 to n do
      begin
         beta[i]:=b[i]/a[i,i];
         s:=-1.0/a[i,i];
         for j:=1 to n do a[i,j]:=s*a[i,j];
      end;
   k:=0;
   for i:=1 to n do b[i]:=beta[i];
   repeat
      emax:=0.0;
      k:=k+1;
      for i:=1 to n do
         begin
            s:=beta[i];
            for j:=1 to n do s:=s+a[i,j]*b[j];
            b[i]:=b[i]+s;
            if b[i] <> 0 then s:=s/b[i];
            if abs(s) > emax then emax:=abs(s);
         end;
   until (emax < er) or (k = kmax);
end;
Procedure SEIDEL_T(var n:integer;a:mat;b:vect;er:real);
{type mat=array[1..n,1..n] of real;vect=array[1..n] of real;}
var i,j,k,kmax:integer;beta:vect;ax:mat;s,emax:real;
begin
   {transformare matrice}
   for i:=1 to n do
      begin
         for j:=1 to i do
            begin
               s:=0.0;
               for k:=1 to n do s:=s+a[k,i]*a[k,j];
               ax[i,j]:=s; ax[j,i]:=s;
            end;
         s:=0;
         for j:=1 to n do s:=s+a[j,i]*b[j];
         beta[i]:=s;
      end;
   {rezolvarea propiu-zisă}
   kmax:=100;
   for i:=1 to n do
      begin
         beta[i]:=b[i]/ax[i,i];
         s:=-1.0/ax[i,i];
         for j:=1 to n do ax[i,j]:=s*ax[i,j];
      end;
   k:=0;
   for i:=1 to n do
      b[i]:=beta[i];
   repeat
      emax:=0.0;
      k:=k+1;
      for i:=1 to n do
         begin
            s:=beta[i];
            for j:=1 to n do
               s:=s+ax[i,j]*b[j];
            b[i]:=b[i]+s;
            if b[i] <> 0 then s:=s/b[i];
            if abs(s) > emax then emax:=abs(s);
         end;
   until (emax < er) or (k = kmax);
end;
```

Fig. 2

MEF-AI.1.2.7 FUNCŢII MATLAB

Pentru obţinerea soluţiei unui sistem de ecuaţii liniare de forma $AX = B$ sau $XA = B$ (v. rel. (2)) se poate folosi una din comenzile X=A\B sau X=B/A.

În practică, ecuaţiile liniare de forma $AX = B$ cu matricea A pătratică sunt mai des întâlnite. Dacă matricea A este singulară (determinantul este nul) atunci soluţia ecuaţiei $AX = B$ este unică. În vederea evaluării preliminare asupra existenţei soluţiei unice în MATLAB se poate calcula rangul matricei extinse a sistemului obţinută prin concatenarea matricei sistemului A cu matricea (vectorul) termenilor liberi B, folosind comanda rank([A B]). Dacă rangul matricei A este egal cu rangul matricei extinse sistemul este determinat cu soluţie unică. În caz contrar sistemul poate fi nedeterminat cu o infinitate de soluţii sau incompatibil.

Pe de altă parte, în multe situaţii, inclusiv a MEF, este important a se evalua preliminar precizia soluţiei obţinute. Astfel, se poate calcula precizia cu relaţia, $p=2{,}2.10^{-15}\text{cond}(A)$ [Ghinea, 2003], în care, cond(A) este comanda MATLAB de calcul a unui factor de condiţionare care este o măsură a apropierii matricei A de singularitate (pentru o matrice singulară factorul de condiţionare este infinit ţi pentru matricea unitate este 1).

Testele arată că varianta cu operatorul / (împărţire la dreapta) este mult mai rapidă, mai ales, pentru sistemele de ecuaţii mari şi foarte mari.

MATLAB foloseşte, pentru matrice pătratice, două metode de bază, pentru factorizarea matricelor:

Factorizarea Cholesky-pentru matrice simetrice, pozitiv definite

Factorizarea Lower Upper (LU)-eliminarea Gauss, pentru matrice pătratice

Toate aceste metode de factorizare se apelează, în MATLAB, utilizând funcţiile specifice: **chol** şi **lu**. Caracteristica, valabilă în toate metodele, o constituie utilizarea matricei triunghiulare ale carei elemente, situate deasupra sau sub diagonala principală, sunt toate egale cu zero.

Factorizarea Cholesky se realizează cu funcţia MATLAB **chol**, care se apelează cu una din sintaxele din tab. 1 care generează matricea superior triunghiulară şi pentru a doua variantă de apelare în plus generează un factor p care în cazul în care are valoarea zero matricea este pozitiv definită. Rezolvarea sistemelor de ecuaţii **AX=B** prin factorizarea Cholesky presupune, X=U\(L\B).

Prin factorizarea **lower-upper (lu)**, o *matrice pătratică* este descompusă sub forma produsului a două matrice triunghiulare: L inferior triunghiulară (lower) permutată şi U, o matrice superior triunghiulară (upper). Rezolvarea sistemelor de ecuaţii **AX=B** prin factorizarea **lower upper** presupune accesarea comenzii X=U\(L\B).

Tab. 1

Nr. crt.	*Funcţia*	*Semnificaţie rezultat*	*Exemple*	*Rezultat*
1	chol	Matricea triunghiulară superior şi factorul de pozitivitate	L=chol(pascal(3)) [L p]=chol(pascal(3)) (L*L'=A)	1 1 1 0 1 2 0 0 1 0
2	lu	Matricele triunghiulare superior şi inferior	[L U]=lu(pascal(3)) (L*U=pascal(3))	1.0000 0 0 1.0000 0.5000 1.0000 1.0000 1.0000 0 1.0000 1.0000 1.0000 0 2.0000 5.0000 0 0 -0.5000

Toate sistemele de ecuaţii liniare, are permit utilizarea matricelor triunghiulare (factorizări), se rezolvă uşor folosind *pre-* sau *post-substituţia (substituţia înainte şi înapoi, retrosubstituţoa)*. Sistemul superior triunghiular se rezolvă prin *substituţie înapoi (retrosubstituţia)* folosind funcţia RS.