Curs 2

Sisteme de ecuaţii liniare

Un sistem de ecuaţii liniare are forma generală:

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

sau matricial:

$A.x=b$ în care $A \in R^{nxn}$, $b \in R^n$

Sistemul admite soluţia unică $x \in R^n$ dacă matricea este inversabilă, caz în care soluţia se exprimă sub forma:

$x=A^{-1}.b$

Metodele de rezolvare :

- *metode exacte* - care furnizează soluţia exactă a sistemului dacă se neglijează erorile de rotunjire.
- *metode aproximative* sau *iterative* - care construiesc un şir , convergent către soluţia exactă a sistemului .
- *Metodele directe* aduc sistemul prin *transformări de echivalenţă*, la un sistem particular (diagonal, triunghiular, etc), care se rezolvă cu mijloace elementare.
- Metodele exacte se bazează pe *factorizare gaussiană* sau pe *factorizare ortogonală*.
- Complexitatea metodelor exacte este $O(n^3)$, motiv care le restrânge aplicabilitatea la rezolvarea sistemelor de ordin nu prea mare ($n<1000$)
- In cazul metodelor aproximative, procesul iterativ de generare a şirului $x^{(k)}$ este oprit la un rang p, în momentul în care $x^{(p)}$ reprezintă o aproximaţie satisfăcătoare a soluţiei .
- Complexitatea metodelor iterative este $O(n^2)$ într-un pas, ele fiind recomandate pentru rezolvarea sistemelor mari ($n>50$), dacă se asigură o convergenţă rapidă..

Metode gaussiene directe

❑ Rezolvarea sistemelor triunghiulare

Presupunem *condiţia de nesingularitate* pentru un sistem triunghiular $a_{ii} \neq 0$

- *sistem superior triunghiular* cu $a_{ij}=0$ pentru $i>j$.

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + \ \ldots & + \ a_{1n}x_n & = & b_1 \\ & & a_{22}x_2 & + \ \ldots & + \ a_{2n}x_n & = & b_2 \\ & & & \ldots & & & \\ & & & & a_{nn}x_n & = & b_n \end{array}$$

- *sistem inferior triunghiular* cu $a_{ij}=0$ pentru $i<j$.

$$\begin{array}{ccccccccc} a_{11}x_1 & & & & & & & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & & & & & = & b_2 \\ \cdots & & \cdots & & \cdots & & & & \cdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array}$$

Sistemul superior triunghiular se rezolvă prin *substituţie înapoi* folosind relaţiile

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} A_{ij} x_j}{A_{ii}}, \qquad i = n : -1 : 1$$

Sistemul inferior triunghiular se rezolvă prin *substituţie înainte* cu relaţiile:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} A_{ij} x_j}{A_{ii}}, \qquad i = 1 : n$$

```
function x = SST1(A, b)
% rezolvare sistem superior triunghiular
% Intrari:   A=matrice sistem
%            b=vector termeni liberi
% Iesiri:    x=vector necunoscute
[n, n] = size(A);
x = zeros(n,1);
x(n) = b(n)/A(n, n);
for i = n-1:-1:1
   x(i)=(b(i)-A(i,i+1:n)*x(i+1:n))/A(i,i);
end
```

Metodă recursivă

Scriem separat ultima linie din sistemul inferior triunghiular:

$$\begin{bmatrix} A_{11} & 0 \\ a_{n1}^T & \alpha_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \xi_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \beta_n \end{bmatrix}$$

Pentru a obţine un algoritm recursiv

```
function x = SITrec(n, A, b)
% rezolvare recursiva sistem inferior triunghiular
if n == 1
  x(1,1)=b(1)/A(1,1);
else
   x(1:n-1,1)=SITrec(n-1, A(1:n-1,1:n-1), b(1:n-1));
   x(n,1) = (b(n,1)-A(n,1:n-1)*x(1:n-1,1))/A(n,n);
end
```

Rezolvare pe coloane

O versiune orientată pe coloane este:

```
function x = SITcol(A, b)
% rezolvare orientata pe coloane a
% sistemului inferior triunghiular
[n,n] = size(A);
x = b;
for k = 1 : n
  x(k) = x(k)/A(k, k);
  x(k+1:n)=x(k+1:n)-x(k)*A(k+1:n,k);
end
```

Eliminare gaussiană.

Teoremă *Dacă* $A^{[p]}=(a_{ij})_{1\le i,j\le n}$, $A^{[p]}\in R^{n\times n}$, *cu* $p=1:n$ *sunt nesingulare, atunci există ,* $T\in R^{n\times n}$ *nesingulară şi inferior triunghiulară astfel încât matricea* $T*A=U$ *este superior triunghiulară.*

Formăm o transformare elementară

$T_p=I_n-t_p*e_p^T$

în care I_n este matricea unitate e_p este coloana p a acesteia, iar t_p, un vector coloană, pe care-l vom numi *vector Gauss*, cu primele componente nule şi celelalte, deocamdată neprecizate:

$t_p=[0\ldots0\ t_{p+1,p}\ldots t_{np}]^T$

Determinăm transformarea T_p astfel încât atunci când se aplică asupra unui vector $x\in R^n$ îi lasă primele p componente nemodificate şi îi anulează restul componentelor:

$T_px=(I_n-t_pe_p^T)x=x-t_p(e_p^Tx)=x-t_px_p$

de unde:

$$(T_p\cdot x)_i = x_i - t_{ip}\cdot x_p = \begin{cases} x_i, & i\le p\\ x_i - t_{ip}x_p, & i>p\end{cases}$$

Impunând anularea ultimelor componente obţinem:

$$t_{ip}=\frac{x_i}{x_p},\quad i=p+1:n$$

Aşadar vectorul Gauss este:

$$t_p=\begin{bmatrix}0 & \cdots & 0 & \dfrac{x_{p+1}}{x_p} & \cdots & \dfrac{x_n}{x_p}\end{bmatrix}^T$$

```
function [t,x]=VecG(p,x)
% determina vectorul Gauss asociat unui vector x,
% caruia ii anuleaza componentele x[p+1:n]
x=x(:);
n=length(x);
t=zeros(n,1);
t(p+1:n)=x(p+1:n)./x(p);
x(p+1:n)=0;
```

Aplicăm transformarea T_p asupra unui vector oarecare $y\in R^n$:

$T_p.y = (I_n - t_p.e_p^T).y = y - t_p.(e_p^T.y) = y - t_p.y_p$

$$(T_p\cdot y)_i = \begin{cases} y_i, & i\le p\\ y_i - \dfrac{x_i}{x_p}\cdot y_p, & i>p\end{cases}$$

```
function y=TG(y,p,t)
% aplica transformarea Gauss unui vector oarecare y
y=y(:); %ii facem vectori
t=t(:);
```

```
n=length(t);
y(p+1:n)=y(p+1:n)-t(p+1:n).*y(p)
```

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & 1 & \cdots & \cdots \\ & & -t_{p+1,p} & \ddots & \\ & & -t_{n,p} & \cdots & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & \cdots & a_{1p} & \cdots & a_{1n} \\ 0 & a_{22} & a_{2p} & \cdots & a_{2n} \\ \cdots & \cdots & a_{pp} & \cdots & \cdots \\ 0 & 0 & a_{p+1,p} & \cdots & a_{pn} \\ 0 & 0 & a_{n,p} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1p} & \cdots & a_{1n} \\ 0 & a_{22} & a_{2p} & \cdots & a_{2n} \\ \cdots & & a_{pp} & \cdots & a_{pn} \\ & & 0 & \ddots & \\ 0 & & 0 & & a_{nn} \end{bmatrix}$$

Considerăm ca vector **x** coloana **p** a matricei **A** (parţial adusă la formă triunghiulară), iar vector **y** vor fi pe rând coloanele **j** ale matricei **A** situate în dreapta coloanei **p (j>p)**

$T_p*A_p=A_{p+1}$

$T_p.A_p = T_p.[a_1 \; ... \; a_p \; ... \; a_n] = [T_p.a_1 \; ...T_p.a_p \; ... \; T_p.a_n]$

Pornind cu matricea **A** pătrată se aplică pe rând o transformare Gauss coloanelor **1,2,… n-1**
Matricea generală de transformare $T=T_{n-1}...T_2T_1$
va determina obţinerea unei matrici transformate **T*A** superior triunghiulară

```
function [A, b] = Gauss(A, b)
% triunghiularizare prin eliminare Gauss
% Intrări :
%   A = matrice sistem
%   b = vector termeni liberi
% Ieşiri :
%   A = matrice sistem superior triunghiular
%   b = termeni liberi sistem triunghiular
[n, n] = size(A);
for  p = 1:n -1
   [t,A(:,p)]=VecG(p,A(:,p));
   for j=p+1:n
      A(:,j)=TG(A(:,j),p,t);
   end
   b=TG(b,p,t);
end
```

Algoritmul poate fi simplificat ţinând cont că transformarea Gauss aplicată vectorului **A(:,p)** nu îi modifică primele **p** componente, deci ar putea fi aplicată vectorului **A(p+1:n, p),** eliminând parametrul **p** din funcţiile **VecG()** şi **TG()** în care **p** devine 1.

```
function [t,x]=VecG(x)
% determina vectorul Gauss asociat
% unui vector x,caruia ii anuleaza
% componentele x[p+1:n]
x=x(:);
n=length(x);
t=zeros(n,1);
t(2:n)=x(2:n)/x(1);
x(2:n)=0;

function y=TG(y,t)
% aplica transformarea Gauss
```

```
% unui vector oarecare y
y=y(:); %ne asiguram ca sunt
t=t(:); %vectori
n=length(t);
y(2:n)=y(2:n)-t(2:n)*y(p)

function [A, b] = Gauss(A, b)
% triunghiularizare sistem
% prin eliminare gaussiana
% Intrari :  A = matrice sistem
%            b = vector termeni liberi
% Iesiri :   A = matrice sistem triunghiular
%            b = termeni liberi sistem triunghiular
[n, n] = size(A);
for  p = 1 : n-1
   for i=p+1 : n
     t = A(i, p) / A(p, p);
     A(i,p:n) = A(i, p:n) - t* A(p, p:n);
     b(i) = b(i) - t * b(p);
  end
end
```

Eliminare gaussiană cu pivotare parţială

Procesul de triunghiularizare eşuează dacă elementul diagonal actualizat este nul, i.e. dacă submatricea a matricei iniţiale este nulă. Chiar pentru valori nenule, dar mici ale acestuia stabilitatea numerică a metodei este afectată.
Strategia de pivotare parţială alege dintre liniile `i=p:n` acea linie `q` pentru care elementul conducător A_{qp} este maxim în valoare absolută.

Eliminare gaussiană cu pivotare totală

Se alege ca element principal pivot, primul element maxim în valoare absolută A_{lm}, din submatricea delimitată de ultimele `n-p+1` linii şi coloane ale matricei A_p, pentru ca acesta să ocupe poziţia `(p,p)` trebuiesc interschimbate liniile `l` şi `p` şi coloanele `m` şi `p`.
Această transformare *total stabilizată* se exprimă prin $A_{p+1}=T_p P_{pl} A_p P_{pm}$

- înmulţirea la stânga cu P_{pl} permută în A_p liniile p şi $l \geq p$
- înmulţirea la dreapta cu P_{pm} permută A_p în coloanele m şi $m \geq p$

$$\left(\underbrace{T_p \cdot P_{pl} \cdot A_p \cdot P_{pm}}_{A_{p+1}}\right) \cdot \left(\underbrace{P_{pm} \cdot x_p}_{x^*}\right) = \left(\underbrace{T_p \cdot P_{pl} \cdot b_p}_{b_{p+1}}\right)$$

- Se observă că în cursul transformării, x este premultiplicat cu P_{pm}, ceeace conduce la permutarea componentelor k şi m. Aceasta impune ţinerea evidenţei schimbărilor de coloane, printr-un vector de indexare a lui x.

Factorizare LU

Dacă $A^{[p]}=(a_{ij})_{1 \leq i,j \leq p}$ cu $p=1:n$ *sunt nesingulare atunci există o matrice* $L \in R^{nxn}$, *nesingulară triunghiular inferioară şi o matrice* $U \in R^{nxn}$, *nesingulară triunghiular superioară astfel ca* $A=LU$.

Sistemul **Ax=b** poate fi rescris **LUx=b** sau

Ly=b
Ux=y

Rezolvarea sistemului se reduce la rezolvarea a două sisteme triunghiulare.
In MATLAB se foloseşte funcţia **[L, U] = lu(A)**

Factorizare directă Crout

$$A_{ij} = \sum_{m=1}^{\min(i,j)} L_{im} U_{mj}$$

$$A_{ip} = \sum_{m=1}^{p-1} L_{im} U_{mp} + L_{ip} U_{pp} \Rightarrow L_{ip} = A_{ip} - \sum_{m=1}^{p-1} L_{im} U_{mp}, \qquad i = p : n$$

$$A_{pj} = \sum_{m=1}^{p-1} L_{pm} U_{mj} + L_{pp} U_{pj} \Rightarrow U_{pj} = \frac{A_{pj} - \sum_{m=1}^{p-1} L_{pm} U_{mj}}{L_{pp}}$$

```
function A = crout(A)
% factorizare directa Crout
% Intrări: A = matricea de factorizat
% Ieşiri:  A = matricele factori
    [m, n] = size(A);
    for p = 1 : n
      for i = p : n
          s = A(i,1:p-1)*A(1:p-1,p);
          A(i,p) = A(i,p) - s;
      end;
      for j = p+1 : n
          s = A(p,1:p-1)*A(1:p-1,j);
          A(p,j) = (A(p,j) - s) / A(p,p);
      end
    end
```

Factorizare Cholesky

Dacă matricea sistemului este:

- *simetrică* ($A_{ij} = A_{ji}$ sau $A = A^T$)
- *pozitiv-definită* ($x^T . A.x>0, \ \forall x \in R^n, \ x \neq 0$)

factorizarea are forma particulară $A=L.L^T=R^T.R$

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}, \qquad i = 1 : n$$

$$L_{ij} = \frac{A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk}}{L_{jj}} \qquad j = 1 : i-1$$

Rezolvarea sistemelor tridiagonale

$$\begin{cases} b_1x_1 + c_1x_2 & = d_1 \\ a_2x_1 + b_2x_2 + c_2x_3 & = d_2 \\ \dots & \\ a_ix_{i-1} + b_ix_i + c_ix_{i+1} & = d_i \\ \dots & \\ a_nx_{n-1} + b_nx_n & = d_n \end{cases}$$

Prin eliminare gaussiană sistemul devine bidiagonal :

$$\begin{cases} b_1x_1 + c_1x_2 & = d_1 \\ b_2x_2 + c_2x_3 & = d_2 \\ \dots & \\ b_ix_i + c_ix_{i+1} & = d_i \\ \dots & \\ b_nx_n & = d_n \end{cases}$$

- Eliminare subdiagonală

$$b_p = b_p - \frac{a_p}{b_{p-1}} \cdot c_{p-1}$$

$$d_p = d_p - \frac{a_p}{b_{p-1}}, \quad p = 2 : n$$

- Rezolvare sistem 2-diagonal

$$x_n = \frac{d_n}{b_n}$$

$$x_i = \frac{d_i - c_ix_{i+1}}{b_i}, \quad i = n - 1 : -1 : 1$$

Rezolvarea sistemelor tridiagonale

```
function d = tridi(a, b, c, d)
% Intrări:   a = subdiagonală
%            b = diagonală principală
%            c = supradiagonală
%            d = termeni liberi
% Ieşiri:    d = necunoscute
%eliminare element subdiagonal
  a = a(:);
  [m, n] = size(A);
  for i = 2 : n
      t = a(i) / b(i-1);
      b(i) = b(i) - t * c(i-1);
      d(i) = d(i) - t * d(i-1);
  end
 % rezolvarea sistemului bidiagonal
  d(n) = d(n) / b(n);
```

```
  for i = n-1 : -1 : 1
    d(i) = (d(i) - c(i) * d(i+1)) / b(i);
  end
```

Inversarea matricelor

- Inversarea matricelor triunghiulare

  $A.B=I_n \quad B=A^{-1}$

- matrice superior triunghiulară

$$b_{jj} = \frac{1}{a_{jj}}, \quad j = 1 : n$$

$$b_{ij} = -\frac{1}{a_{ii}} \cdot \sum_{k=i+1}^{j} a_{ik} b_{kj}, \quad i = j-1 : -1 : 1$$

- matrice inferior triunghiulară

$$b_{ii} = \frac{1}{a_{ii}}, \quad i = 1 : n$$

$$b_{ij} = -\frac{1}{a_{ii}} \cdot \sum_{k=j}^{i-1} a_{ik} b_{kj}, \quad j = 1 : i-1$$

Inversarea matricelor triunghiulare

```
function B = invTrSup(A)
 % inversare matrice superior-triunghiulara
 % Intrări: A=matrice superior triunghiulară
 % Ieşiri:  B = inversa matricii A
 [n, n] = size(A);
 for j = n : -1 : 1
   B(j, j) = 1 / A(j, j);
   for i = j - 1 : -1 : 1
     B(i,j)=-A(i,i+1:j)*B(i+1:j,j)/A(i,i);
   end
 end
 B = triu(B);
```

Metoda Gauss-Jordan

$B=[A \mid I_n]$

- normalizare: $A_{pj} = \dfrac{A_{pj}}{A_{pp}}, \quad p = 1 : n, \; j = p : 2n$

- reducere: $A_{ij} = A_{ij} - A_{ip}A_{pj}, \; p = 1 : n, \; i = 1 : n, \; j = p : 2n$

```
function X = invGauss(A)
% inversare matrice prin rezolvare
% simultana a n sisteme
% Intrări:A = matricea de inversat
% Ieşiri: X = matricea inversă
[m, n] = size(A);
X=eye(n);
```

```
% triunghiularizare sisteme
for p = 1 : n-1
  for i = p+1 : n
     t = A(i,p) / A(p,p);
     A(i,p) = 0;
     A(i,p+1:n) = A(i,p+1:n)-t * A(p,p+1:n)
  end
  X(i,1:n) = X(i,1:n) - t * X(p,1:n);
end
% rezolvare sisteme triunghiulare
for i = n : -1 : 1
  for k = 1 : n
     suma = A(i,i+1:n)*X(i+1:n,k);
     X(i,k) = (X(i,k) - suma) / A(i,i)
  end
end
```