

## Laborator 2

# Rezolvarea sistemelor liniare determinate. Sisteme triunghiulare. Eliminarea gaussiană

### 2.1 Tema

Rezolvarea numerică a sistemelor liniare inferior și superior triunghiulare prin metode de substituție directă precum și rezolvarea sistemelor liniare determinate dense prin metode de transformare cum este eliminarea gaussiană cu diferite strategii de pivotare.

### 2.2 Preliminarii

Un sistem de  $n$  ecuații liniare cu  $n$  necunoscute, scris în formă matriceală

$$Ax = b, \quad (2.1)$$

unde matricea  $A = (a_{ij})_{i=1:m, j=1:n} \in \mathbb{R}^{n \times n}$  și vectorul  $b = (b_i)_{i=1:n} \in \mathbb{R}^n$  sunt date, iar  $x = (x_j)_{j=1:n} \in \mathbb{R}^n$  este vectorul necunoscutelor, se numește sistem liniar determinat.

Sistemul (2.1) admite a soluție unică pentru toți  $b \in \mathbb{R}^n$  dacă și numai dacă matricea  $A$  este nesingulară, i.e. inversabilă. Într-un astfel de caz unica soluție a sistemului (2.1) se scrie

$$x = A^{-1}b. \quad (2.2)$$

Trebuie să subliniem că formula (2.2) *nu este recomandată* pentru calculul soluției sistemului  $Ax = b$ , după cum nu sunt recomandate și alte metode "bine cunoscute" cum este regula lui Cramer etc.

Vom prezenta o categorie de metode fiabile și eficiente de rezolvare a sistemelor determinate și anume așa numitele *metode directe*, bazate pe reducerea, printr-o secvență finită de transformări elementare, la rezolvarea, prin simplă substituție, a unui sau a două sisteme triunghiulare. În această categorie intră metode precum eliminarea gaussiană sau factorizarea LU.

### 2.2.1 Rezolvarea sistemelor triunghiulare

O matrice  $L \in \mathbb{R}^{n \times n}$ , cu  $l_{ij} = 0$  pentru toți  $i < j$  se numește *inferior triunghiulară*. O matrice  $U \in \mathbb{R}^{n \times n}$ , cu  $u_{ij} = 0$  pentru toți  $i > j$  se numește *superior triunghiulară*. Sistemul  $Ax = b$  se numește inferior (superior) triunghiular dacă matricea  $A$  este inferior (superior) triunghiulară. O matrice triunghiulară este nesingulară dacă și numai dacă toate elementele sale diagonale sunt nenule.

Algoritmii pentru rezolvarea sistemelor triunghiulare sunt foarte simpli, întrucât necunoscutele pot fi calculate, într-o ordine definită, prin substituție numerică directă. Considerăm, mai întâi, un sistem inferior triunghiular

$$Lx = b,$$

unde  $L \in \mathbb{R}^{n \times n}$ , cu  $l_{ij} = 0$  pentru  $i < j$ ,  $l_{ii} \neq 0$ ,  $i = 1 : n$ , și  $b \in \mathbb{R}^n$ . Prima ecuație este

$$l_{11}x_1 = b_1,$$

de unde

$$x_1 = b_1/l_{11}. \quad (2.3)$$

În general, dacă știm  $x_1, x_2, \dots, x_{i-1}$ , putem rezolva ecuația  $i$

$$\sum_{j=1}^{i-1} l_{ij}x_j + l_{ii}x_i = b_i$$

și obținem

$$x_i = \left( b_i - \sum_{k=1}^{i-1} l_{ik}x_k \right) / l_{ii}. \quad (2.4)$$

Formulele (2.3) și (2.4) definesc un algoritm pentru calculul soluției sistemului  $Lx = b$ , cunoscut sub denumirea de metoda *substituției înainte*.

**ALGORITM 2.1 (S-*INF-TR*)** (Dată o matrice  $L \in \mathbb{R}^{n \times n}$ , nesingulară, inferior triunghiulară și un vector  $b \in \mathbb{R}^n$ , algoritmul calculează soluția  $x$  a sistemului  $Lx = b$ .)

1. **pentru**  $i = 1 : n$ 
  1.  $s \leftarrow b_i$
  2. **pentru**  $j = 1 : i - 1$ 
    1.  $s \leftarrow s - l_{ij}x_j$
  3.  $x_i \leftarrow s/l_{ii}$

Acest algoritm necesită  $N_{S\_INF\_TR} \approx n^2$  flopi și  $M_{S\_INF\_TR} \approx n^2/2$  locații de memorie. Dacă vectorul  $b$  nu mai este necesar, soluția  $x$  poate suprascrie vectorul  $b$  pe măsură ce este calculată.

Pentru a rezolva un sistem liniar superior triunghiular nesingular

$$Ux = b,$$

unde  $U$  este o matrice superior triunghiulară de ordinul  $n$ , cu  $u_{ii} \neq 0$ ,  $i = 1 : n$ , și  $b \in \mathbb{R}^n$ , observăm că ultima ecuație este

$$u_{nn}x_n = b_n,$$

din care avem

$$x_n = b_n/u_{nn}. \quad (2.5)$$

Mai mult, dacă  $x_n, x_{n-1}, \dots, x_{i+1}$ , sunt deja calculate, atunci, din ecuația  $i$

$$u_{ii}x_i + \sum_{j=i+1}^n u_{ij}x_j = b_i$$

obținem

$$x_i = \left( b_i - \sum_{j=i+1}^n u_{ij}x_j \right) / u_{ii}. \quad (2.6)$$

Din formulele (2.5) și (2.6), care definesc așa numita *substituție înapoi*, obținem următorul algoritm.

ALGORITM 2.2 (*S\\_SUP\\_TR*) (Dată o matrice  $U \in \mathbb{R}^{n \times n}$ , superior triunghiulară, nesingulară și un vector  $b \in \mathbb{R}^n$ , acest algoritm calculează soluția  $x$  a sistemului  $Ux = b$ .)

1. **pentru**  $i = n : -1 : 1$ 
  1.  $s \leftarrow b_i$
  2. **pentru**  $j = i + 1 : n$ 
    1.  $s \leftarrow s - u_{ij}x_j$
  3.  $x_i \leftarrow s/u_{ii}$

Ca și în cazul inferior triunghiular, efortul de calcul este dat de  $N_{S\_SUP\_TR} = n^2$  flopi și memoria necesară este  $M_{S\_SUP\_TR} \approx n^2/2$  locații.

## 2.3 Eliminarea gaussiană

Eliminarea gaussiană este o tehnică pentru reducerea sistemului  $Ax = b$  la un sistem superior triunghiular echivalent

$$MAx = Mb,$$

i.e. cu matricea  $U = MA$  superior triunghiulară. Matricea de transformare  $M$  este o matrice inferior triunghiulară unitară obținută ca o secvență (produs) de transformări (matrice) inferior triunghiulare elementare

$$M = M_{n-1}M_{n-2}\dots M_1.$$

O matrice inferior triunghiulară elementară (ITE) de ordin  $n$  și indice  $k$  este o matrice de forma

$$M_k = I_n - m_k e_k^T, \quad m_k = \underbrace{[0 \ \dots \ 0]}_{k-1} \mu_{kk} \ \mu_{k+1,k} \ \dots \ \mu_{nk}]^T.$$

Triangularizarea de mai sus există dacă toate submatricele lider principale  $A^{[k]} = A(1 : k, 1 : k)$  sunt nesingulare, iar procesul de triangularizare are loc conform schemei de calcul

### 1. pentru $k = 1 : n - 1$

1. Se determină matricea ITE  $M_k$ , i.e. scalarii  $\mu_{ik}$ ,  $i = k + 1 : n$ , astfel încât  $(M_k A)_{ik} = 0$ ,  $i = k + 1 : n$

2.  $A \leftarrow M_k A$

Scalarii  $\mu_{ik}$ , numiți *multiplicatori gaussiani*, care asigură satisfacerea condiției de la 1.1 au expresia

$$\mu_{ik} = a_{ik}/a_{kk}, \quad i = k + 1 : n,$$

și pot fi memorați în locul zerourilor create. Operația  $A \leftarrow M_k A$  se execută eficient dacă ținem seama că primele  $k - 1$  coloane ale lui  $A$  nu sunt afectate, coloana  $k$  este afectată într-un mod cunoscut, iar coloanele  $a_j$ ,  $j \in k + 1 : n$  suferă transformarea

$$(M_k a_j)_i = ((I_n - m_k e_k^T) a_j)_i = (a_j - m_k a_{kj})_i = a_{ij} - \mu_{ik} a_{kj}, \quad i = k + 1 : n.$$

Cele prezentate mai sus definesc algoritmul  $G$  de eliminare gaussiană descris mai jos.

**ALGORITM 2.3 ( $G$ )** (Dată o matrice  $A \in \mathbb{R}^{n \times n}$  cu submatricele lider principale  $A^{[k]}$ ,  $k = 1 : n - 1$ , nesingulare, acest algoritm suprascrie triunghiul superior al lui  $A$ , inclusiv diagonala, cu matricea superior triunghiulară  $U = M_{n-1}M_{n-2}\dots M_1 A$ . Triunghiul inferior al lui  $A$  este suprascris cu multiplicatorii gaussiani  $\mu_{ik}$  care definesc matricele ITE  $M_k$ ,  $k = 1 : n - 1$ .)

1. **pentru**  $k = 1 : n - 1$ 
  1. **pentru**  $i = k + 1 : n$ 
    1.  $a_{ik} \leftarrow \mu_{ik} = a_{ik}/a_{kk}$
  2. **pentru**  $i = k + 1 : n$ 
    1. **pentru**  $j = k + 1 : n$ 
      1.  $a_{ij} \leftarrow a_{ij} - \mu_{ik}a_{kj}$

Efortul de calcul este  $N_G \approx \frac{2n^3}{3}$  flopi, în timp ce memoria necesară este  $M_G = n^2$  locații.

*Observație.* Nesingularitatea submatricelor lider principale ale matricei  $A$  nu este o condiție necesară pentru existența și unicitatea soluției unui sistem de forma  $Ax = b$ . Pentru a elimina această condiție vom introduce pivotarea.

### 2.3.1 Strategii de pivotare

Singularitatea submatricelor lider principale  $A^{[k]}$  este echivalentă cu anularea elementului  $a_{kk}$ , numit pivot, la pasul  $k$  al algoritmului G. Ca urmare, calculul multiplicatorilor gaussieni nu se poate face. Într-un astfel de caz, este necesară aducerea pe poziția pivotului a unui element nenul, preferabil de modul cât mai mare, prin permutare de linii și/sau coloane.

Întrucât interschimbarea a două linii ale unei matrice este echivalentă cu pre-multiplicarea acelei matrice cu o matrice de permutare elementară iar interschimbarea a două coloane este echivalentă cu postmultiplicarea matricei cu o matrice de permutare elementară, întregul proces de "pivotare" poate fi descris în termeni matriceali într-unul din două următoare moduri.

#### Pivotare parțială

Pivotarea parțială are loc prin permutarea numai a liniilor. La pasul  $k$  al algoritmului G se aduce în poziția  $(k, k)$  a pivotului cel mai mare element în modul dintre elementele subdiagonale din coloana  $k$ , fie acesta  $a_{i_k k} \neq 0$ , prin permutarea liniilor  $k$  și  $i_k$ . Acest lucru este echivalent cu premultiplicarea matricei  $A$  cu matricea de permutare elementară  $P_{i_k k} \stackrel{\text{not}}{=} P_k$ , astfel încât pasul  $k$  realizează

$$A \leftarrow M_k P_k A,$$

iar întregul algoritm realizează triangularizarea matricii  $A$  prin secvența

$$A \leftarrow U = M_{n-1} P_{n-1} M_{n-2} P_{n-2} \dots M_1 P_1 A.$$

Algoritmul corespunzător este dat mai jos.

ALGORITM 2.4 (*GPP – eliminarea gaussiană cu pivotare parțială*)  
 (Dată o matrice  $A \in \mathbb{R}^{n \times n}$ , algoritmul suprascrie triunghiul superior al lui  $A$  cu matricea superior triunghiulară  $U = M_{n-1}P_{n-1} \dots M_1P_1A$ . Triunghiul strict inferior al lui  $A$  este suprascris de multiplicatorii gaussieni  $\mu_{ik}$  ce definesc matricele ITE  $M_k$ ,  $k = 1 : n - 1$ , iar întregii  $i_k$ , definind permutările de linii, sunt memorati într-un vector  $p$ , astfel încât  $p(k) = i_k$ , pentru  $k = 1 : n - 1$ .)

1. **pentru**  $k = 1 : n - 1$ 
  1. Se determină primul  $i_k \in k : n$  astfel încât  $|a_{i_k k}| = \max_{i=k:n} |a_{ik}|$ .
  2.  $p(k) \leftarrow i_k$
  3. **pentru**  $j = k : n$ 
    1.  $a_{kj} \leftrightarrow a_{i_k j}$
  4. **pentru**  $i = k + 1 : n$ 
    1.  $a_{ik} \leftarrow \mu_{ik} = a_{ik}/a_{kk}$
  5. **pentru**  $i = k + 1 : n$ 
    1. **pentru**  $j = k + 1 : n$ 
      1.  $a_{ij} \leftarrow a_{ij} - \mu_{ik}a_{kj}$

Complexitatea algoritmului GPP este aceeași cu cea a algoritmului G.

### Pivotare completă

Proprietăți numerice mai bune se obțin dacă pivotul de la pasul  $k$  se alege drept cel mai mare element în modul dintre elementele  $a_{ij}$ , cu  $i = k : n$ ,  $j = k : n$ . Acest cel mai mare element în modul, fie el  $a_{i_k j_k}$ , este adus în poziția  $(k, k)$  a pivotului prin permutarea liniilor  $k$  și  $i_k$  și a coloanelor  $k$  și  $j_k$ . Acest lucru este echivalent cu premultiplicarea matricei  $A$  cu matricea de permutare elementară  $P_{i_k k} \stackrel{\text{not}}{=} P_k$  și cu postmultiplicarea cu matricea de permutare elementară  $Q_{j_k k} \stackrel{\text{not}}{=} Q_k$ , astfel încât pasul  $k$  realizează

$$A \leftarrow M_k P_k A Q_k,$$

iar întregul algoritm realizează triangularizarea matricii  $A$  prin secvența

$$A \leftarrow U = M_{n-1}P_{n-1} \dots M_2P_2M_1P_1AQ_1Q_2 \dots Q_{n-1}.$$

Algoritmul corespunzător, de eliminare gaussiană cu pivotare completă, este dat mai jos.

ALGORITM 2.5 (*GPC – eliminarea gaussiană cu pivotare completă*)  
 (Dată a matrice  $A \in \mathbb{R}^{n \times n}$ , acest algoritm suprascrie triunghiul superior al lui  $A$  cu matricea superior triunghiulară  $U$ . Triunghiul strict inferior al lui  $A$  este suprascris cu multiplicatorii gaussieni  $\mu_{ik}$ ,  $k = 1 : n - 1$ ,  $i = k + 1 : n$ . Întregii  $i_k$  și  $j_k$ , ce definesc permutările de linii, respectiv coloane, sunt memorati în doi vectori  $p$  și  $q$ , astfel încât  $p(k) = i_k$  și  $q(k) = j_k$ , pentru  $k = 1 : n - 1$ .)

1. **pentru**  $k = 1 : n - 1$ 
  1. Se determină  $i_k \in k : n$  și  $j_k \in k : n$  astfel încât
 
$$|a_{i_k j_k}| = \max_{i=k:n, j=k:n} |a_{ij}|.$$
  2.  $p(k) \leftarrow i_k$
  3.  $q(k) \leftarrow j_k$
  4. **pentru**  $j = k : n$ 
    1.  $a_{kj} \leftrightarrow a_{i_k j}$
  5. **pentru**  $i = 1 : n$ 
    1.  $a_{ik} \leftrightarrow a_{ij_k}$
  6. **pentru**  $i = k + 1 : n$ 
    1.  $a_{ik} \leftarrow \mu_{ik} = a_{ik}/a_{kk}$
  7. **pentru**  $i = k + 1 : n$ 
    1. **pentru**  $j = k + 1 : n$ 
      1.  $a_{ij} \leftarrow a_{ij} - \mu_{ik} a_{kj}$

Complexitatea algoritmului GPC este aceeași cu cea a algoritmului G.

## 2.4 Rezolvarea sistemelor liniare determinate

Așa cum s-a precizat, algoritmi GPP și GPC stau la baza unei metode de rezolvare a sistemelor liniare nesingulare  $Ax = b$ . Deși din punct de vedere numeric algoritmul GPC este mai bun, GPP este suficient de bun pentru necesitățile practice curente și, de aceea, este folosit pe scară largă.

Admițând că desfacem matricea  $A$  rezultată din algoritmi GPP și GPC în matricele componente: matricea superior triunghiulară  $U$  și matricea strict inferior triunghiulară  $M$  a multiplicatorilor gaussiani, introducem următoarele sintaxe de utilizare a algoritmilor GPP și GPC

$$[M, U, p] = \text{GPP}(A) \quad [M, U, p, q] = \text{GPC}(A).$$

Considerăm sistemul liniar  $Ax = b$ , cu  $A$  nesingulară. Evident, sistemul  $Ax = b$  este echivalent cu sistemul

$$M_{n-1}P_{n-1} \dots M_1P_1Ax = M_{n-1}P_{n-1} \dots M_1P_1b,$$

unde  $M_k, P_k$  sunt matricele utilizate de GPP, i.e. cu sistemul superior triunghiular

$$Ux = M_{n-1}P_{n-1} \dots M_1P_1b,$$

Deci, pentru a rezolva  $Ax = b$ , trebuie să calculăm vectorul

$$b \leftarrow d = M_{n-1}P_{n-1} \dots M_1P_1b,$$

folosind următoarea schemă:

1. **pentru**  $k = 1 : n - 1$

1.  $b \leftarrow P_k b$
2.  $b \leftarrow M_k b.$

Algoritmul corespunzător este dat mai jos.

ALGORITM 2.6 (*SL-GPP – Rezolvarea sistemelor liniare cu GPP*)  
(Dată o matrice nesingulară  $A \in \mathbb{R}^{n \times n}$  și un vector  $b \in \mathbb{R}^n$ , acest algoritim calculează soluția  $x \in \mathbb{R}^n$  a sistemului liniar  $Ax = b$ , folosind eliminarea gaussiană cu pivotare parțială.)

1.  $[M, U, p] = \text{GPP}(A)$
2. **pentru**  $k = 1 : n - 1$ 
  1.  $b_k \leftrightarrow b_{p(k)}$
  2. **pentru**  $i = k + 1 : n$ 
    1.  $b_i \leftarrow b_i - \mu_{ik} b_k$
3.  $x = \text{S\_SUP\_TR}(U, b)$

Complexitatea algoritmului este dată de

$$N_{SL\_GPP} = N_{GPP} + \sum_{k=1}^{n-1} 2(n-k) + N_{S\_SUP\_TR} \approx \frac{2n^3}{3} + n^2 + n^2 \approx \frac{2n^3}{3} \approx N_{GPP}.$$

În cazul utilizării algoritmului GPC sistemul  $Ax = b$  este echivalent cu sistemul

$$M_{n-1}P_{n-1} \dots M_1P_1AQ_1 \dots Q_{n-1}Q_{n-1} \dots Q_1x = M_{n-1}P_{n-1} \dots M_1P_1b.$$

Notând

$$Q_{n-1} \dots Q_1x = y,$$

rezultă că  $y$  poate fi calculat ca soluție a sistemului superior triunghiular  $Uy = d$ , unde  $d = M_{n-1}P_{n-1} \dots M_1P_1b$ . Din expresia lui  $y$  avem

$$x = Q_1Q_2 \dots Q_{n-1}y$$

care poate fi calculat cu schema

1.  $x \leftarrow y$
2. **pentru**  $k = n - 1 : -1 : 1$ 
  1.  $x \leftarrow Q_k x.$

Rezultă algoritmul.

ALGORITM 2.7 (*SL-GPC – rezolvarea sistemelor liniare cu GPC*)  
(Dată o matrice nesingulară  $A \in \mathbb{R}^{n \times n}$  și un vector  $b \in \mathbb{R}^n$ , acest algoritim calculează soluția  $x \in \mathbb{R}^n$  a sistemului liniar  $Ax = b$ , folosind eliminarea gaussiană cu pivotare completă.)



1.  $[M, U, p, q] = \text{GPC}(A)$
2. **pentru**  $k = 1 : n - 1$ 
  1.  $b_k \leftrightarrow b_{p(k)}$
  2. **pentru**  $i = k + 1 : n$ 
    1.  $b_i \leftarrow b_i - \mu_{ik} b_k$
3.  $x = \text{S\_SUP\_TR}(U, b)$
4. **pentru**  $k = n - 1 : -1 : 1$ 
  1.  $x_k \leftrightarrow x_{q(k)}$

## 2.5 Sarcini de lucru

### 2.5.1 A. In laborator

1. Se elaborează și editează programe, sub forma unor funcții MATLAB, pentru implementarea algoritmilor de rezolvare a sistemelor liniare inferior și superior triunghiulare.
2. Se testează corectitudinea algoritmilor și programelor pentru sisteme triunghiulare cu coeficienți aleatori de 7 și 25 de ecuații.
3. Se elaborează și editează un program, sub forma unei funcții MATLAB, pentru implementarea algoritmului G de triangularizare prin eliminare gaussiană și se testează corectitudinea algoritmului și programului pentru matrice cu elemente aleatoare de ordinul 7 și 25. Ce se întâmplă dacă înainte de apelul funcției execuția instrucțiunea  $A(1, 1:2) = A(2, 1:2)$ ?
4. Se elaborează și editează un program, sub forma unei funcții MATLAB, pentru implementarea algoritmului GPP de triangularizare prin eliminare gaussiană cu pivotare parțială și se testează corectitudinea algoritmului și programului pentru matrice cu elemente aleatoare de ordinul 7 și 25.
5. Se elaborează și editează un program, sub forma unei funcții MATLAB, pentru implementarea algoritmului de rezolvare a sistemelor liniare  $Ax = b$  folosind programul GPP de triangularizare prin eliminare gaussiană cu pivotare parțială și se testează corectitudinea algoritmului și programului pentru sisteme liniare de 7 și 25 de ecuații cu 7, respectiv 25 de necunoscute.

### 2.5.2 B. Acasă

1. Se elaborează și editează programe, sub forma unor funcții MATLAB, pentru implementarea algoritmilor de rezolvare a sistemelor liniare inferior și superior bidiagonale și se testează corectitudinea lor pentru sisteme bidiagonale cu coeficienți aleatori de 100 și 500 de ecuații.

10 *LABORATOR 2. REZOLVAREA SISTEMELOR LINIARE DETERMINATE*

- 2 Se elaborează și editează un program, sub forma unei funcții MATLAB, pentru implementarea algoritmului GPC de triangularizare prin eliminare gaussiană cu pivotare parțială și se testează corectitudinea algoritmului și programului pentru matrice cu elemente aleatoare de ordinul 100 și 500.
- 3 Se elaborează și editează un program, sub forma unei funcții MATLAB, pentru implementarea algoritmului de rezolvare a sistemelor liniare  $Ax = b$  folosind programul GPC de triangularizare prin eliminare gaussiană cu pivotare completă și se testează corectitudinea algoritmului și programului pentru sisteme liniare de 100 și 500 de ecuații cu 100, respectiv 500 de necunoscute.