

Operații cu matrice în MATLAB. Rezolvarea eficientă a sistemelor de ecuații liniare. Factorizări LU. Inversarea matricelor prin partiționare.

Cuprins

1	Obiective laborator	1
2	Noțiuni teoretice	1
2.1	Complexitatea regulii Cramer	1
2.2	Factorizarea LU	2
2.2.1	Descompunerea matricei în L și U	2
2.2.2	Rezolvarea sistemelor triunghiulare	5
2.3	Inversarea matricilor prin partiționare	6
2.3.1	Matrice blocuri	6
2.3.2	Complementul lui Schur	7
2.3.3	Calcularea inversei	7
3	Probleme	8

1 Obiective laborator

În urma parcurgerii acestui laborator, studentul va fi capabil să:

- factorizeze o matrice folosind una dintre metodele LU: Crout, Doolittle, Cholesky;
- rezolve recursiv un sistem triunghiular;

2 Noțiuni teoretice

2.1 Complexitatea regulii Cramer

La liceu, sistemele de ecuații se rezolvau folosind *regula lui Cramer*. În continuare, demonstrăm de ce această abordare este ineficientă computațional.

Fie un sistem de n ecuații cu n necunoscute, $Ax = b$, cu $\det(A) \neq 0$. Folosind regula lui Cramer urmează să calculăm n determinanți pentru fiecare necunoscută, înlocuind pe rând coloane din A cu b . În total, calculăm $n + 1$ determinanți.

Determinantul unei matrici se poate calcula în două moduri:

$$\det(A) = \sum_{i=1}^n (-1)^{i+j} a_{ij} \det(A_{ij}), \quad \forall j \in \{1, 2, \dots, n\} \quad (1)$$

$$\det(A) = \sum_{j=1}^n (-1)^{i+j} a_{ij} \det(A_{ij}), \quad \forall i \in \{1, 2, \dots, n\} \quad (2)$$

Cu A_{ij} am notat matricea de dimensiune $(n-1) \times (n-1)$ rezultată din suprimarea (eliminarea) liniei i și a coloanei j . Cu alte cuvinte, pentru calcularea unui determinant de ordin n trebuie să calculăm n determinanți de ordinul $n-1$. În total, pentru calcularea determinantului de ordin n , am avea:

$$n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1 = n! \quad (3)$$

În cazul unui sistem de n ecuații cu n necunoscute, conform (3), avem o complexitate de ordinul $O(n!)$. Mai departe vom explora metode mai eficiente de calcul, de ordinul $O(n^3)$.

2.2 Factorizarea LU

Factorizarea (sau descompunerea) unei matrici are o aplicabilitate importantă în analiza numerică. Pentru rezolvarea sistemelor liniare, vom discuta despre factorizările LU și QR.

Factorizarea LU presupune descompunerea unei matrici pătratice A într-un produs de două matrici, L și U , unde L este o matrice inferior triunghiulară, iar U este o matrice superior triunghiulară. Astfel, putem scrie $A = LU$. Drept urmare, sistemul de ecuații $Ax = b$ se transformă în două sisteme:

$$Ly = b$$

$$Ux = y$$

Aceste sisteme se numesc *triunghiulare* și se pot rezolva în $O(n^2)$. Așadar, avem 2 pași de făcut:

1. Descompunerea matricii A în L și U ;
2. Rezolvarea sistemelor triunghiulare.

2.2.1 Descompunerea matricii în L și U

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

Prin efectuarea descompunerii direct, ar rezulta un sistem cu n^2 ecuații și $n^2 + n$ necunoscute. Putem ”scăpa” de necunoscutele în plus folosind mai multe metode, cele mai cunoscute fiind *Crout*, *Doolittle* și *Cholesky*.

Metoda Crout

Metoda Crout presupune ca toate elementele de pe diagonală matricii U să fie egale cu 1. Astfel, pentru o matrice de dimensiune 3×3 , putem scrie sistemul de ecuații:

$$\begin{array}{lll} l_{11} = a_{11} & l_{11}u_{12} = a_{12} & l_{11}u_{13} = a_{13} \\ l_{21} = a_{21} & l_{21}u_{12} + l_{22} = a_{22} & l_{21}u_{13} + l_{22}u_{23} = a_{23} \\ l_{31} = a_{31} & l_{31}u_{12} + l_{32} = a_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} = a_{33} \end{array}$$

Algoritmul în MATLAB pentru Crout poate fi gândit astfel:

1. Folosim un indice p cu care ne ”plimbăm” pe coloane;
2. Observăm că pentru fiecare coloană avem două seturi de ecuații:
 - Din primele $p - 1$ ecuații putem calcula $u_{ip}, \forall j \in \{1, 2, \dots, p - 1\}$;
 - Din restul, putem calcula $l_{ip}, \forall j \in \{p, p + 1, \dots, n\}$.

Algorithm 1 Metoda Crout

```

1:  $n \leftarrow$  numărul de linii a matricei  $A$ 
2:  $L \leftarrow$  matricea 0 de dimensiune  $n \times n$ 
3:  $U \leftarrow$  matricea identitate de dimensiune  $n \times n$ 
4: for  $p = 1$  to  $n$  do
5:   for  $i = 1$  to  $p - 1$  do
6:      $U(i, p) \leftarrow \frac{A(i, p) - L(i, 1:i) \cdot U(1:i, p)}{L(i, i)}$ 
7:   end for
8:   for  $i = p$  to  $n$  do
9:      $L(i, p) \leftarrow A(i, p) - L(i, 1:i) \cdot U(1:i, p)$ 
10:  end for
11: end for

```

Metoda Doolittle

Metoda Doolittle presupune ca toate elementele de pe diagonala matricei L să fie egale cu 1. Astfel, pentru o matrice 3×3 , putem scrie sistemul de ecuații:

$$\begin{array}{lll}
 u_{11} = a_{11} & u_{12} = a_{12} & u_{13} = a_{13} \\
 l_{21}u_{11} = a_{21} & l_{21}u_{12} + u_{22} = a_{22} & l_{21}u_{13} + u_{23} = a_{23} \\
 l_{31}u_{11} = a_{31} & l_{31}u_{12} + l_{32}u_{22} = a_{32} & l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33}
 \end{array}$$

Algoritmul în MATLAB pentru Doolittle poate fi gândit astfel:

1. Folosim un indice p cu care ne ”plimbăm” pe linii;
2. Observăm că pentru fiecare linie avem două seturi de ecuații:
 - Din primele $p - 1$ ecuații putem calcula $l_{pi}, \forall i \in \{1, 2, \dots, p - 1\}$;
 - Din restul, putem calcula $u_{pi}, \forall i \in \{p, p + 1, \dots, n\}$.

Algorithm 2 Metoda Doolittle

```

1:  $n \leftarrow$  numărul de linii a matricei  $A$ 
2:  $L \leftarrow$  matricea identitate de dimensiune  $n \times n$ 
3:  $U \leftarrow$  matricea 0 de dimensiune  $n \times n$ 
4: for  $p = 1$  to  $n$  do
5:   for  $i = 1$  to  $p - 1$  do
6:      $L(p, i) \leftarrow \frac{A(p, i) - L(p, 1:(i-1)) \cdot U(1:(i-1), i)}{U(i, i)}$ 
7:   end for
8:   for  $i = p$  to  $n$  do
9:      $U(p, i) \leftarrow A(p, i) - L(p, 1:(i-1)) \cdot U(1:(i-1), i)$ 
10:  end for
11: end for

```

Metoda Cholesky

Descompunerea Cholesky se remarcă prin faptul că matricea U este setată ca fiind transpusa (sau hermitica) matricei L , adică $A = LL^*$.

Fie o matrice A simetrică de dimensiune $n \times n$.

$$A \text{ pozitiv - definită} \Leftrightarrow \mathbf{x}^* A \mathbf{x} > 0, \forall \mathbf{x} \in \mathbb{C}^n \setminus \{\mathbf{0}\}$$

În același mod se poate defini și conceptul de matrice *negativ-definită*, înlocuind semnul $>$ cu $<$.

Descompunerea Cholesky se poate aplica doar pe matrici simetrice, pozitiv-definite.

Demonstrație. Fie A o matrice oarecare.

$$A = LL^* \implies A^* = LL^* \implies A = A^*$$

Demonstrăm acum că A trebuie să fie pozitiv-definită. Fie $Ax = b$:

$$\begin{aligned} Ax &= b \\ x^* Ax &= x^* b \\ x^* LL^* x &= x^* b, \quad y := L^* x \\ y^* y &= x^* b \end{aligned}$$

Cum $y^* y > 0 \quad \forall y \neq 0$, rezultă că $x^* b > 0$ și deci $x^* Ax > 0$. ■

Revenind, pentru o matrice 3×3 , putem scrie sistemul de ecuații:

$$\begin{array}{lll} l_{11}^2 = a_{11} & l_{11}l_{21} = a_{12} & l_{11}l_{31} = a_{13} \\ l_{11}l_{21} = a_{21} & l_{21}^2 + l_{22}^2 = a_{22} & l_{21}l_{31} + l_{22}l_{32} = a_{23} \\ l_{11}l_{31} = a_{31} & l_{21}l_{31} + l_{22}l_{32} = a_{32} & l_{31}^2 + l_{32}^2 + l_{33}^2 = a_{33} \end{array}$$

Algoritmul în MATLAB pentru Cholesky poate fi gândit astfel:

1. Folosim un indice p cu care ne "plimbăm" pe linii și un alt indice i cu care ne "plimbăm" pe coloane;
2. Observăm că avem 2 tipuri de ecuații:
 - $i = p$: Putem calcula $l_{pp} = \sqrt{a_{pp} - \sum_{j=1}^i l_{pj}^2}$;
 - $i \neq p$: Putem calcula $l_{pi} = \frac{a_{pi} - \sum_{j=1}^i l_{pj}l_{ij}}{l_{ii}}$.
3. Pentru că matricea A este simetrică, putem ignora partea de deasupra diagonalei principale a sistemului.
4. Se observă că cele două sume sunt echivalente atunci când $p = i$.

Algorithm 3 Metoda Cholesky

```

1:  $n \leftarrow$  numărul de linii a matricei  $A$ 
2:  $L \leftarrow$  matrice 0 de dimensiune  $n \times n$ 
3: for  $p = 1$  to  $n$  do
4:   for  $i = 1$  to  $p$  do
5:      $s \leftarrow L(p, 1 : i) \cdot L(i, 1 : i)^T$ 
6:     if  $i = p$  then
7:        $L(p, p) \leftarrow \sqrt{A(p, p) - s}$ 
8:     else
9:        $L(p, i) \leftarrow \frac{A(p, i) - s}{L(i, i)}$ 
10:    end if
11:  end for
12: end for
```

2.2.2 Rezolvarea sistemelor triunghiulare

Sistemele triunghiulare pot fi de 2 tipuri, *superioare* sau *inferioare*, în funcție de tipul matricei. Tratăm prima data cazul sistemelor superior triunghiulare.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{nn}x_n &= b_n \end{aligned}$$

Pentru a rezolva un sistem superior triunghiular, putem folosi metoda *substituției înapoi*:

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}}, \quad \forall i \in \{n, n-1, \dots, 1\} \quad (4)$$

Algorithm 4 Substituție înapoi pentru sisteme triunghiulare superioare

```

1:  $n \leftarrow$  numărul de linii a matricei  $A$ 
2:  $x \leftarrow$  vector plin de 0 de dimensiune  $n$ 
3: for  $i = n$  to 1 step  $-1$  do
4:    $x(i) \leftarrow \frac{b(i) - A(i, (i+1):n) \cdot x((i+1):n)}{A(i, i)}$ 
5: end for

```

Sistemele inferior triunghiulare arată similar:

$$\begin{aligned}
 a_{11}x_1 &= b_1 \\
 a_{21}x_1 + a_{22}x_2 &= b_2 \\
 &\vdots \\
 a_{n1}x_1 + \dots + a_{nn}x_n &= b_n
 \end{aligned}$$

Pentru a rezolva un sistem inferior triunghiular, putem folosi metoda *substituției înainte*:

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}}, \quad \forall i \in \{1, 2, \dots, n\} \quad (5)$$

Algorithm 5 Substituție înainte pentru sisteme triunghiulare inferioare

```

1:  $n \leftarrow$  numărul de linii a matricei  $A$ 
2:  $x \leftarrow$  vector plin de 0 de dimensiune  $n$ 
3: for  $i = 1$  to  $n$  do
4:    $x(i) \leftarrow \frac{b(i) - A(i, 1:(i-1)) \cdot x(1:(i-1))}{A(i, i)}$ 
5: end for

```

2.3 Inversarea matricilor prin partiționare

Partiționarea matricilor este o tehnică folosită pentru a simplifica operațiile cu matrice. Dacă partiționăm bine, atunci putem paraleliza operațiile pe blocuri, scăzând astfel timpul de execuție.

2.3.1 Matrice blocuri

Fie matricea $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$. Aceasta poate fi împărțită, de exemplu, în 4 blocuri de dimensiune 2×2 , $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, unde

$$A_{11} = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \quad A_{12} = \begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix} \quad A_{21} = \begin{bmatrix} 9 & 10 \\ 13 & 14 \end{bmatrix} \quad A_{22} = \begin{bmatrix} 11 & 12 \\ 15 & 16 \end{bmatrix}$$

Pentru matrici conforme, operațiile de adunare și înmulțire se pot face pe blocuri. De exemplu, pentru adunare:

$$A + B = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} \\ A_{21} + B_{21} & A_{22} + B_{22} \end{bmatrix}$$

Iar pentru înmulțire:

$$AB = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

2.3.2 Complementul lui Schur

Complementul lui Schur [1] apare atunci când pe o matrice bloc aplicăm eliminarea Gaussiană. Fie matricea $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$. Pentru a elimina elementele de sub diagonala principală (matricea C) astfel:

$$\begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix}$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} I & 0 \\ -D^{-1}C & I \end{bmatrix} = \begin{bmatrix} A - BD^{-1}C & B \\ 0 & D \end{bmatrix}$$

Astfel, complementul lui Schur este definit:

$$M/A := D - CA^{-1}B, \text{ dacă } A \text{ inversabilă}$$

$$M/D := A - BD^{-1}C, \text{ dacă } D \text{ inversabilă}$$

2.3.3 Calcularea inversei

Continuăm cu eliminarea Gaussiană pentru a calcula inversa unei matrici.

$$\begin{bmatrix} I & -B(M/A)^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & (M/A) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & (M/A) \end{bmatrix}$$

$$\begin{bmatrix} A^{-1} & 0 \\ 0 & (M/A)^{-1} \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & (M/A) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

Astfel, inversa va fi (dacă și M/A este inversabilă):

$$\begin{aligned} M^{-1} &= \begin{bmatrix} A^{-1} & 0 \\ 0 & (M/A)^{-1} \end{bmatrix} \begin{bmatrix} I & -B(M/A)^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} & 0 \\ 0 & (M/A)^{-1} \end{bmatrix} \begin{bmatrix} I + B(M/A)^{-1}CA^{-1} & -B(M/A)^{-1} \\ -CA^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} + A^{-1}B(M/A)^{-1}CA^{-1} & -A^{-1}B(M/A)^{-1} \\ -(M/A)^{-1}CA^{-1} & (M/A)^{-1} \end{bmatrix} \end{aligned}$$

Dacă D și M/D este inversabilă atunci putem scrie

$$M^{-1} = \begin{bmatrix} (M/D)^{-1} & -(M/D)^{-1}BD^{-1} \\ -D^{-1}C(M/D)^{-1} & D^{-1} + D^{-1}C(M/D)^{-1}BD^{-1} \end{bmatrix}$$

Dacă și A și D sunt inversabile, atunci, egalând cele două matrice pentru inversă, putem ajunge la următoarea factorizare:

$$M^{-1} = \begin{bmatrix} (M/D)^{-1} & 0 \\ 0 & (M/A)^{-1} \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ -CA^{-1} & I \end{bmatrix}$$

3 Probleme

1. Pentru matricea dată mai jos, determinați matricele L și U folosind metoda Crout.

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 11 \\ 3 & 22 & 42 \end{bmatrix}$$

2. Pentru matricea de la exercițiul anterior aplicați factorizarea Doolittle.

3. Pentru matricea $A = \begin{bmatrix} 14 & 8 & 3 \\ 8 & 5 & 2 \\ 3 & 2 & 1 \end{bmatrix}$ verificați dacă este pozitiv definită folosind teorema cercurilor Gershgorin. Aplicați factorizarea Cholesky.

4. Scrieți două funcții în MATLAB care să rezolve un sistem de ecuații superior triunghiular, respectiv inferior triunghiular. Folosiți următoarele prototipuri:

```
function x = superior(U, b)
function x = inferior(U, b)
```

5. Scrieți o funcție în MATLAB pentru fiecare din cele 3 descompuneri studiate.

6. Calculați inversa pentru matricea $A = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$.

Referințe

- [1] J. Schur. Über potenzreihen, die im innern des einheitskreises beschränkt sind. *Journal für die reine und angewandte Mathematik*, 147:205–232, 1917.