

# Eliminare gaussiană cu pivotare totală și scalare. Algoritmul Thomas pentru rezolvarea sistemului 3-diagonal.

Colaboratori: Andrei STAN, Florin Pop, Mihaela Andreea-Vasile

February 17, 2025

## Cuprins

<b>1</b>	<b>Obiective laborator</b>	<b>1</b>
<b>2</b>	<b>Noțiuni teoretice</b>	<b>1</b>
2.1	Eliminarea gaussiană - G	1
2.2	Eliminarea gaussiană cu pivotare parțială - GPP	2
2.3	Eliminarea gaussiană cu pivotare parțială cu pivot scalat - GPPS	3
2.4	Eliminarea gaussiană cu pivotare totală - GPT	4
2.5	Algoritmul Thomas pentru rezolvarea sistemului 3-diagonal	5
<b>3</b>	<b>Probleme rezolvate</b>	<b>6</b>
3.1	Problema 1	6
3.2	Problema 2	6
3.3	Problema 3	7
3.4	Problema 4	8
3.5	Problema 5	9
<b>4</b>	<b>Probleme</b>	<b>10</b>
4.1	Problema 1	10
4.2	Problema 2	10
4.3	Problema 3	10
4.4	Problema 4	10

# 1 Obiective laborator

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

- Transforme o matrice nesingulară într-o matrice superior sau inferior triunghiulară folosind una din metodele prezentate de eliminare gaussiană;
- Implementeze algoritmi de eliminare gaussiană prezentați;
- Implementeze algoritmul Thomas.

# 2 Noțiuni teoretice

Acest laborator are ca scop familiarizarea cu metodele numerice directe de transformare a unei matrice nesingulare la forma superior triunghiulară sau inferior triunghiulară. Aceste metode poartă numele de *eliminări Gaussiene*. Cea mai simplă metodă de eliminare este *eliminarea gaussiană cu pivotare parțială*. Pentru această metodă vom prezenta algoritmul GPP. Cea mai bună metodă este *eliminarea gaussiană cu pivotare totală sau completă*, numit GPT în cadrul acestui laborator. O tehnică de scalare care micșorează eroarea și elimină anularea flotantă este *pivotarea parțială cu pivot scalat pe coloană* (algoritmul GPPS).

Eliminarea gaussiană este echivalentă cu metoda factorizării LU care trebuie să fie utilizată împreună cu o strategie de pivotare adecvată (factorizarea LUP). O strategie de pivotare este necesară în general deoarece este posibil ca eliminarea gaussiană să nu poată transforma o matrice dată la o formă triunghiulară. Pentru o matrice simetrică și pozitiv definită cea mai bună metodă de aducere la forma triunghiulară este folosirea factorizării Cholesky.

Algoritmul Thomas este o formă simplificată a eliminării gaussiene pentru o matrice tridiagonală. Acest algoritm este folosit pentru rezolvarea sistemelor de ecuații liniare tridiagonale (des întâlnite în metodele de interpolare cu funcții spline). Complexitatea algoritmului Thomas este  $O(n)$  în timp ce eliminările gaussiene au complexitatea  $O(n^3)$ .

## 2.1 Eliminarea gaussiană - G

Eliminarea gaussiană este o tehnică pentru transformarea matricei  $A$  la forma superior triunghiulară. Matricea de transformare  $T$  este o matrice inferior triunghiulară unitară obținută ca o secvență (produs) de transformări inferior triunghiulare elementare de forma  $T = T_{n-1}T_{n-2} \dots T_1$ , unde matricele  $T_p$  sunt inferior triunghiulare, de ordin  $n$ , de forma:

$$T_p = I_n - t_p e_p^T,$$

$e_p$  este coloana  $p$  din matricea unitate și

$$t_p = \begin{bmatrix} 0 & \dots & 0 & \mu_{p+1p} & \dots & \mu_{np} \end{bmatrix}.$$

Metoda de mai sus se poate realiza dacă toate submatricele de forma  $A^{[p]} = A(1:p, 1:p)$  sunt nesingulare. Scalarii  $\mu_{ip}$ , numiți *multiplicatori gaussieni*, ce asigură satisfacerea condiției de anulare a elementelor din coloana  $p$ , de sub diagonala principală, au expresia:

$$\mu_{ip} = a_{ip}/a_{pp}, \quad i = p+1:n$$

În efectuarea operației  $A \leftarrow T_p A$ , se vor memora multiplicatorii gaussieni în locul zerourilor create sub diagonala principală, primele  $p-1$  coloane ale lui  $A$  nu sunt afectate, iar coloanele  $a_j$ ,  $j = p+1:n$  sunt transformate astfel:

$$(T_p a_j)_i = ((I_n - t_p e_p^T) a_j)_i = (a_j - t_p a_{pj})_i = a_{ij} - \mu_{ip} a_{pj}, \quad i = p+1:n.$$

Algoritmul G de eliminare gaussiană este:

---

**Algorithm 1** Eliminare gaussiană
 

---

```

procedure G( $A$ )
  for  $p = 1 : n - 1$  do
    for  $i = p + 1 : n$  do
       $\mu_{ip} = a_{ip}/a_{pp}$ ;
       $a_{ip} = 0$ ;
      for  $j = p + 1 : n$  do
         $a_{ij} = a_{ij} - \mu_{ip}a_{pj}$ ;
      end for
    end for
  end for
  return  $A$ ;
end procedure

```

---

Numărul de operații pentru algoritmul G este  $O(n^3)$  (aproximativ  $\frac{2n^3}{3}$  operații), iar memoria folosită, conform cu schema descrisă este  $O(n^2)$ . Nesingularitatea submatricelor nu este o condiție necesară pentru existența și unicitatea soluției unui sistem de forma  $Ax = b$ , unde  $A$  este adusă prin transformare la forma superior triunghiulară. Pentru a elimina această condiție se introduc strategiile de pivotare:

- Eliminarea gaussiană cu pivotare parțială - GPP;
- Eliminarea gaussiană cu pivotare parțială cu pivot scalat - GPPS;
- Eliminarea gaussiană cu pivotare totală - GPT.

Singularitatea submatricelor  $A^{[p]}$  este echivalentă cu anularea elementului  $a_{pp}$ , numit pivot, la pasul  $p$  al algoritmului G. Consecința acestei anulări conduce la imposibilitatea calculului multiplicatorilor gaussieni.

Este necesară aducerea pe poziția pivotului (linia  $p$  și coloana  $p$ ) a unui element nenul, preferabil de modul cât mai mare, prin permutare de linii și/sau coloane.

O matrice de permutare este o matrice care are un singur element nenul egal cu 1 pe orice linie și pe orice coloană a sa. Ea se obține din matricea unitate prin interschimbarea (o dată sau de mai multe ori) a două linii și/sau a două coloane. Iată un exemplu (interschimbarea liniilor 1 și 2, apoi a coloanelor 2 și 3):

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Interschimbarea a două linii ale unei matrice este echivalentă cu multiplicarea acelei matrice cu o matrice de permutare la stânga. Interschimbarea a două coloane este echivalentă cu multiplicarea matricei cu o matrice de permutare la dreapta.

## 2.2 Eliminarea gaussiană cu pivotare parțială - GPP

Pivotarea parțială are loc numai prin permutarea liniilor. La pasul  $p$  al algoritmului G se aduce în poziția  $(p, p)$  a pivotului cel mai mare element în modul dintre elementele subdiagonale din coloana  $p$ , fie acesta  $a_{i_p p} \neq 0$ , prin permutarea liniilor  $p$  și  $i_p$ . Acest lucru este echivalent cu multiplicarea matricei  $A$  la stânga cu matricea de permutare  $P_{i_p p} \stackrel{\text{not}}{=} P_p$ , astfel încât pasul  $p$  calculează  $A \leftarrow T_p P_p A$ , întregul algoritm fiind:

$$A \leftarrow U = T_{n-1} P_{n-1} T_{n-2} P_{n-2} \dots T_1 P_1 A$$

Algoritmul GPP de eliminare gaussiană cu pivotare parțială este:

---

**Algorithm 2** Eliminarea gaussiană cu pivotare parțială
 

---

```

1: procedure GPP( $A$ )
2:   for  $p = 1 : n - 1$  do
3:     Determină primul  $i_p$  ( $p \leq i_p \leq n$ ) a.î.  $|a_{i_p p}| = \max_{i=p:n} \{|a_{ip}|\}$ ;  $v(p) = i_p$ ;
4:      $v(p) = i_p$ ;
5:     for  $j = p : n$  do
6:        $a_{pj} \leftrightarrow a_{i_p j}$ ;
7:     end for
8:     for  $i = p + 1 : n$  do
9:        $\mu_{ip} = a_{ip} / a_{pp}$ ;
10:       $a_{ip} = 0$ ;
11:      for  $j = p + 1 : n$  do
12:         $a_{ij} = a_{ij} - \mu_{ip} a_{pj}$ ;
13:      end for
14:    end for
15:  end for
16:  return  $A, v$ ;
17: end procedure

```

---

Pașii 8-14 reprezintă algoritmul G pentru matricea permutată. Vectorul  $v$  memorează permutările de linii.

### 2.3 Eliminarea gaussiană cu pivotare parțială cu pivot scalat - GPPS

Această tehnică este similară cu precedentă, doar că definim la început un factor de scalare pentru fiecare linie  $i$ , factor de forma:

$$s_i = \max_{j=p:n} \{|a_{ij}|\} \quad \text{sau} \quad s_i = \sum_{j=p}^n |a_{ij}|$$

Dacă există un  $i$  astfel încât  $s_i = 0$  atunci matricea este singulară. Pașii următori vor stabili inter-schimbările care se vor face. La pasul  $p$  se va găsi întregul  $i_p$  astfel încât:

$$\frac{|a_{i_p p}|}{s_{i_p}} = \max_{i=p:n} \frac{|a_{ip}|}{s_i}$$

Scalarea ne garantează că cel mai mare element din fiecare coloană are înainte de comparațiile necesare pentru schimbare mărimea relativă 1. Scalarea se realizează doar în comparații, nu efectiv în matrice, astfel că împărțirea cu factorul de scalare nu produce nici o eroare de rotunjire.

Algoritmul GPPS de eliminare gaussiană cu pivotare parțială cu pivot scalat este:

**Algorithm 3** Eliminarea gaussiană cu pivotare parțială cu pivot scalat

---

```

1: procedure GPPS( $A$ )
2:   for  $p = 1 : n - 1$  do
3:     Determină  $i_p$  a.î.  $\frac{|a_{i_p p}|}{s_{i_p}} = \max_{i=p:n} \frac{|a_{i p}|}{s_i}$ ;  $v(p) = i_p$ ;
4:     for  $j = 1 : n$  do
5:        $a_{pj} \leftrightarrow a_{i_p j}$ ;
6:     end for
7:     for  $i = p + 1 : n$  do
8:        $\mu_{ip} = a_{ip} / a_{pp}$ ;
9:        $a_{ip} = 0$ ;
10:      for  $j = p + 1 : n$  do
11:         $a_{ij} = a_{ij} - \mu_{ip} a_{pj}$ ;
12:      end for
13:    end for
14:  end for
15:  return  $A, v$ ;
16: end procedure

```

---

Pașii 7-13 reprezintă algoritmul G pentru matricea permutată. Vectorul  $v$  memorează permutările de linii. Complexitatea algoritmului GPP este aceeași cu a lui G.

## 2.4 Eliminarea gaussiană cu pivotare totală - GPT

Stabilitate numerică mai bună se obține dacă pivotul de la pasul  $p$  se alege drept cel mai mare element în modul dintre elementele  $a_{ij}$ , cu  $i = p : n$ ,  $j = p : n$ , fie el  $a_{i_p j_p}$ , și este adus în poziția  $(p, p)$  a pivotului prin permutarea liniilor  $p$  și  $i_p$  și a coloanelor  $p$  și  $j_p$ . Acest lucru este echivalent cu multiplicarea matricei  $A$  la stânga cu matricea de permutare  $P_{i_p p}^{st} \stackrel{\text{not}}{=} P_p^{st}$  și cu multiplicarea matricei  $A$  la dreapta cu matricea de permutare  $P_{j_p p}^{dr} \stackrel{\text{not}}{=} P_p^{dr}$ . La pasul  $p$  se calculează  $A \leftarrow T_p P_p^{st} A P_p^{dr}$ , întregul algoritm fiind:

$$A \leftarrow U = T_{n-1} P_{n-1}^{st} T_{n-2} P_{n-2}^{st} \dots T_1 P_1^{st} A P_1^{dr} P_2^{dr} \dots P_{n-1}^{dr}.$$

Algoritmul GPPS de eliminare gaussiană cu pivotare totală este:

**Algorithm 4** Eliminarea gaussiană cu pivotare totală

---

```

1: procedure GPT( $A$ )
2:   for  $p = 1 : n - 1$  do
3:     Determină  $i_p$  și  $j_p$  ( $p \leq i_p, j_p \leq n$ ) a.i.  $|a_{i_p j_p}| = \max_{i=p:n, j=p:n} \{|a_{ij}|\}$ ;  $v_{st}(p) = i_p$ ;  $v_{dr}(p) = j_p$ ;
4:     for  $j = p : n$  do
5:        $a_{pj} \leftrightarrow a_{i_p j}$ ;
6:     end for
7:     for  $i = 1 : n$  do
8:        $a_{ip} \leftrightarrow a_{i j_p}$ ;
9:     end for
10:    for  $i = p + 1 : n$  do
11:       $\mu_{ip} = a_{ip} / a_{pp}$ ;
12:       $a_{ip} = 0$ ;
13:      for  $j = p + 1 : n$  do
14:         $a_{ij} = a_{ij} - \mu_{ip} a_{pj}$ ;
15:      end for
16:    end for
17:  end for
18:  return  $A, v_{st}, v_{dr}$ ;
19: end procedure

```

---

Pașii 10-16 reprezintă algoritmul G pentru matricea permutată. Vectorul  $v_{st}$  memorează permutările de linii iar  $v_{dr}$  memorează permutările de coloane. Complexitatea algoritmului GPP este aceeași cu a lui G.

## 2.5 Algoritmul Thomas pentru rezolvarea sistemului 3-diagonal

Sistemul tridiagonal se poate scrie compact sub forma  $a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$ ,  $i = 1 : n$  cu mențiunea că  $a_1 = 0$  și  $c_n = 0$ , iar componentele  $x_0$  și  $x_{n+1}$  nu sunt definite. Forma generală este:

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

Algoritmul presupune modificarea coeficienților sistemului și aducerea lor la o formă prin care sistemul de poate rezolva direct prin substituție înapoi.

$$c'_i = \begin{cases} \frac{c_i}{b_i} & i = 1 \\ \frac{c_i}{b_i - c'_{i-1} a_i} & i = 2 : n - 1 \end{cases}$$

$$d'_i = \begin{cases} \frac{d_i}{b_i} & i = 1 \\ \frac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i} & i = 2 : n \end{cases}$$

Prin formulele de mai sus s-a executat de fapt un pas al eliminării gaussiene. Soluția sistemului va fi dată

de formulele:

$$\begin{cases} x_n = d'_n \\ x_i = d'_i - c'_i x_{i+1} \quad i = n-1 : 1 \end{cases}$$

Sistemul nu se mai păstrează în memorie prin întreaga lui matrice, ci doar prin trei vectori de valori corespunzătoare celor trei diagonale.

Algoritmul Thomas este utilizat deoarece este rapid și apare frecvent în practică: interpolări, ecuații diferențiale, etc. Deși situația este rară, algoritmul poate fi instabil dacă  $b_i - c'_i x_{i+1} = 0$  sau numeric zero pentru orice  $i$ . Aceasta se întâmplă dacă matricea este singulară, dar în cazuri rare se poate întâmpla și pentru o matrice nesingulară. Condiția de stabilitate este,  $\forall i$ :

$$|b_i| > |a_i| + |c_i|$$

condiție care indică diagonal-dominanța matricei  $A$ . Dacă algoritmul este numeric instabil se pot aplica strategii de pivotare, ca în cazul eliminării gaussiene.

## 3 Probleme rezolvate

### 3.1 Problema 1

Fie sistemul de ecuații:  $\begin{cases} 5.2 \cdot x_1 + 7.1 \cdot x_2 = 19.8 \\ 2.4 \cdot x_1 + 3.2 \cdot x_2 = 4.1 \end{cases}$  Rezolvați sistemul folosind eliminare gaussiană cu pivotare parțială.

*Soluție:*

Valoarea maximă a coeficienților de pe prima coloană este 5.2 (coeficientul de pe prima linie). Așadar, nu schimbăm ordinea ecuațiilor.

$$\mu_{21} = \frac{2.4}{5.2} = 0.46154$$

Sistemul inițial este echivalent cu:  $\begin{cases} 5.2 \cdot x_1 + 7.1 \cdot x_2 = 19.8 \\ -0.07692 \cdot x_2 = -5.0385 \end{cases}$

În final, obținem soluția:  $\begin{cases} x_1 = -85.625 \\ x_2 = 65.5 \end{cases}$

### 3.2 Problema 2

Să se scrie un program OCTAVE pentru a implementa algoritmul de eliminarea gaussiană.

*Soluție:*

```
function [A, b] = G(A, b)
[n n] = size(A);

for p = 1 : n - 1
    for i = p + 1 : n
        if A(p, p) == 0
            continue;
```

```
endif

tp = A(i, p)/A(p, p);
A(i, p) = 0;
for j = p + 1 : n
    A(i, j) = A(i, j)-tp*A(p, j);
endfor

b(i) = b(i)-tp*b(p);
endfor
endfor
endfunction
```

### 3.3 Problema 3

Să se scrie un program OCTAVE pentru a implementa algoritmul de eliminarea gaussiană cu pivotare parțială.

*Soluție:*

```
function [A b] = GPP(A, b)
[n n] = size(A);

for p = 1 : n - 1
    pivot = -inf;
    linie_pivot = -1;

    %calculez maximul intre elementele A(p : n, p)
    for i = p : n
        if pivot < abs(A(i, p));
            pivot = abs(A(i, p));
            linie_pivot = i;
        endif
    endfor

    %permutarea liniilor linie_pivot si p
    if p ~= linie_pivot
        for j = p : n
            t = A(p, j);
            A(p, j) = A(linie_pivot, j);
            A(linie_pivot, j) = t;
        endfor

        %permutarea elementelor b(linie_pivot) si b(p)
        t = b(linie_pivot);
        b(linie_pivot) = b(p);
        b(p) = t;
    endif

    %eliminare gaussiana
    for i = p + 1 : n
        if A(p, p) == 0
            continue;
        endif

        tp = A(i, p)/A(p, p);
```



```
A(i, p) = 0;
for j = p + 1 : n
    A(i, j) = A(i, j) - tp * A(p, j);
endfor

b(i) = b(i) - tp * b(p);
endfor
endfor
endfunction
```

### 3.4 Problema 4

Să se scrie un program OCTAVE pentru a implementa algoritmul de eliminarea gaussiană cu pivotare totală.

*Soluție:*

```
function [A, b] = GPT(A, b)
[n n] = size(A);

for p = 1 : n - 1
    pivot = -inf;
    linie_pivot = -1;
    coloana_pivot = -1;

    %calculez maximul in submatricea A(p : n, p : n)
    for i = p : n
        for j = p : n
            if pivot < abs(A(i, j));
                pivot = abs(A(i, j));
                linie_pivot = i;
                coloana_pivot = j;
            endif
        endfor
    endfor

    if p ~= linie_pivot
        %permutarea liniilor linie_pivot si p
        for j = p : n
            t = A(p, j);
            A(p, j) = A(linie_pivot, j);
            A(linie_pivot, j) = t;
        endfor

        %permutarea elementelor b(linie_pivot) si b(p)
        t = b(linie_pivot);
        b(linie_pivot) = b(p);
        b(p) = t;
    endif

    %permutarea coloanelor coloana_pivot si p
    if p ~= coloana_pivot
        for i = 1 : n
            t = A(i, p);
            A(i, p) = A(i, coloana_pivot);
            A(i, coloana_pivot) = t;
        endfor
    endif
endfor
```

```
        endfor
    endif

    %eliminare gaussiană
    for i = p + 1 : n
        if A(p, p) == 0
            continue;
        endif

        tp = A(i, p)/A(p, p);
        A(i, p) = 0;
        for j = p + 1 : n
            A(i, j) = A(i, j)-tp*A(p, j);
        endfor

        b(i) = b(i)-tp*b(p);
    endfor
endfor
endfunction
```

### 3.5 Problema 5

Să se scrie un program OCTAVE pentru a implementa algoritmul Thomas de rezolvare a unui sistem 3-diagonal.

*Soluție:*

```
function x = Thomas(a, b, c, d)
    n = length(d);

    % Operariile la limita;
    c(1) = c(1)/b(1);
    d(1) = d(1)/b(1);

    % calculul coeficientilor pe caz general.
    for i = 2 : n-1
        temp = b(i)-a(i)*c(i-1);
        c(i) = c(i)/temp;
        d(i) = (d(i)-a(i)*d(i-1))/temp;
    endfor
    d(n) = (d(n)-a(n)*d(n-1))/(b(n)-a(n)*c(n-1));

    % Substitutia inapoi pentru rezolvarea sistemului de ecuatii
    x(n) = d(n);
    for i = n-1 : -1 : 1
        x(i) = d(i)-c(i)*x(i+1);
    endfor
endfunction
```

**Date de intrare:**

$$a = [2 \ 9 \ 2 \ 3 \ 6], b = [12 \ 15 \ 2 \ 9 \ 1 \ 0], c = [10 \ 3 \ 9 \ 1 \ 4], d = [1 \ 5 \ 9 \ 11 \ 13 \ 7]$$

**Date de ieșire:**

$$x = [0.160494 \quad -0.092593 \quad 2.022634 \quad 0.643118 \quad 1.166667 \quad 2.475995]$$

## 4 Probleme

### 4.1 Problema 1

Implementați în OCTAVE algoritmul GPPS.

### 4.2 Problema 2

Modificați algoritmi GPP și GPT pentru a lucra cu o matrice superior Hessemberg. Implementați H\_GPP și H\_GPT.

### 4.3 Problema 3

Construiți o variantă modificată pentru algoritmul lui Thomas care să lucreze cu matricea:

$$A = \begin{bmatrix} b_1 & 0 & c_1 & & & & 0 \\ 0 & b_2 & 0 & c_2 & & & \\ a_3 & 0 & b_3 & 0 & c_3 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & a_{n-2} & 0 & b_{n-2} & 0 & c_{n-2} \\ & & & a_{n-1} & 0 & b_{n-1} & 0 \\ 0 & & & & a_n & 0 & b_n \end{bmatrix}$$

Scrieți o funcție OCTAVE cu semnătura `function x = solve(a,b,c,d)` care să rezolvă sistemul de ecuații  $Ax = d$ .

### 4.4 Problema 4

Fie sistemul de ecuații: 
$$\begin{cases} 1.5 \cdot x_1 - 2.1 \cdot x_2 = 8.3 \\ -7.6 \cdot x_1 + 3.11 \cdot x_2 = 6.7 \end{cases}$$
 Rezolvați sistemul folosind eliminare GPP, GPPS, GPT.