METODE NUMERICE: Laborator #4
Eliminare gaussiană cu pivotare totală şi scalare. Algoritmul Thomas pentru rezolvarea sistemului 3-diagonal

Titulari curs: *Florin Pop, George-Pantelimon Popescu*

Responsabil Laborator: **Florin Pop**

Obiective Laborator

Acest laborator are ca scop familiarizarea cu metodele numerice directe de transformare a unei matrici nesingulare la forma superior triunghiulară sau inferior triunghiulară. Aceaste metode poartă numele de *eliminări Gaussiene*. Cea mai simplă metodă de eliminare este *eliminarea gaussiană cu pivotare parţială*. 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 micşorează eroarea şi elimină anularea flotantă este *pivotarea parţială 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 şi 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 matrici tridiagonale. Acest algoritm este folosit pentru rezolvarea sistemelor de ecuaţii liniare tridiagonale (des întâlnite în metodele de interpolare cu funcţii spine). Complexitatea algoritmului Thomas este $O(n)$ în timp ce eliminările gaussiene au complexitatea $O(n^3)$.

Eliminarea gaussiana - G

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

$$T_p = I_n - t_p e_p^T \tag{1}$$

unde e_p este coloana p din matricea unitate şi:

$$t_p = \begin{bmatrix} 0 & \cdots & 0 & \mu_{pp} & \cdots & \mu_{np} \end{bmatrix} \tag{2}$$

Metoda de mai sus se poate realiza dacă toate sub-matricele de forma $A^{[p]} = A(1:p, 1:p)$ sunt nesingulare. Scalarii μ_{ip}, numiţi *multiplicatori gaussieni*, care asigură satisfacerea condiţiei de anulare a elementelor din coloana p de sub diagonala principală au expresia:

$$\mu_{ip} = a_{ip}/a_{pp}, \quad i = p+1:n \tag{3}$$

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

$$(T_p a_j)_i = ((I_n - t_p e_p^T)a_j)_i = (a_j - t_p a_{pj})_i = a_{ij} - \mu_{ip} a_{pj}, \quad i = p+1:n. \tag{4}$$

Algoritmul G de eliminare gaussiană este:

Algorithm 1 Eliminare Gaussiană

```
1:  procedure G(A)
2:      for p = 1 : n − 1 do
3:          for i = p + 1 : n do
4:              a_ip ← μ_ip = a_ip/a_pp;
5:          end for
6:          for i = p + 1 : n do
7:              for j = p + 1 : n do
8:                  a_ij ← a_ij − μ_ip a_pj;
9:              end for
10:         end for
11:     end for
12:     Return A;
13: end procedure
```

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

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

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

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

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

Eliminarea gaussiana cu pivotare partiala - GPP

Pivotarea parţială are loc numai prin permutarea liniilor. La pasul p al algoritmului G se aduce în poziţia (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 şi i_p. Acest lucru este echivalent cu multiplicarea matricei A la stânga cu matricea de permutare $P_{i_p p} \overset{\text{not}}{=} P_p$, astfel încât pasul p calculează $A \leftarrow T_p P_p A$, întregul algoritm fiind:

$$A \leftarrow U = T_{n-1} P_{n-1} T_{n-2} P_{n-2} \dots T_1 P_1 A \tag{5}$$

Algoritmul GPP de eliminare gaussiană cu pivotare parţială este:

Algorithm 2 Eliminarea gaussiană cu pivotare parţială

```
1:  procedure GPP(A)
2:      for p = 1 : n − 1 do
3:          Determină primul i_p (p ≤ i_p ≤ n) a.i |a_{i_p p}| = max_{i=p:n} {a_ip};
4:          v(p) = i_p;
5:          for j = p : n do
6:              a_pj ↔ a_{i_p j};
7:          end for
8:          for i = p + 1 : n do
9:              a_ip ← μ_ip = a_ip / a_pp;
10:         end for
11:         for i = p + 1 : n do
12:             for j = p + 1 : n do
13:                 a_ij ← a_ij − μ_ip a_pj;
14:             end for
15:         end for
16:     end for
17:     Return A, v;
18: end procedure
```

Paşii 8-15 reprezintă algoritmul G pentru matricea permutată. Vectorul v memorează permutările de linii. Complexitatea algoritmului GPP este aceeaşi cu a lui G.

Eliminarea gaussiana cu pivotare partiala cu pivot scalat - GPPS

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

$$s_i = \max_{j=1:n} \{|a_{ij}|\} \quad sau \quad s_i = \sum_{j=1}^{n} |a_{ij}| \tag{6}$$

Dacă există un i astfel încât $s_i = 0$ atunci matricea este singulară. Paşii următori vor stabili interschimbările care se vor face. La pasul p se va găsi cel mai mic întreg i_p astfel încât:

$$\frac{|a_{i_p p}|}{s_{i_p}} = \max_{j=1:n} \frac{|a_{jp}|}{s_j} \tag{7}$$

Scalarea ne garantează că cel mai mare element din fiecare coloana are înainte de comparaţiile necesare pentru scimbare mărimea relativă 1. Scalarea se realizează doar în comparaţii, nu efectiv în matrice, astfel că impărţirea cu factorul de scalare nu produce nici o eroare de rotunjire.

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

Algorithm 3 Eliminarea gaussiană cu pivotare parţială cu pivot scalat

```
1: procedure GPPS(A)
2:     for p = 1 : n − 1 do
3:         Determină i_p a.i. |a_{i_p p}|/s_{i_p} = max_{j=1:n} |a_{jp}|/s_j; v(p) = i_p;
4:         for j = 1 : n do
5:             a_{pj} ↔ a_{i_p j};
6:         end for
7:         for i = p + 1 : n do
8:             a_{ip} ← μ_{ip} = a_{ip}/a_{pp};
9:         end for
10:        for i = p + 1 : n do
11:            for j = p + 1 : n do
12:                a_{ij} ← a_{ij} − μ_{ip} a_{pj};
13:            end for
14:        end for
15:    end for
16:    Return A, v;
17: end procedure
```

Paşii 8-15 reprezintă algoritmul G pentru matricea permutată. Vectorul v memorează permutările de linii. Complexitatea algoritmului GPP este aceeaşi cu a lui G.

Eliminarea gaussiana cu pivotare totala - GPT

Stabilitate numerică mai bună se obţine 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 $a_{i_p j_p}$, şi este adus în poziţia (p, p) a pivotului prin permutarea liniilor p şi i_p şi a coloanelor p şi j_p. Acest lucru este echivalent cu multiplicarea matricei A la stânga cu matricea de permutare $P^{st}_{i_p p} \overset{\text{not}}{=} P^{st}_p$ şi cu multiplicarea matricei A la dreapta cu matricea de permutare $P^{dr}_{j_p p} \overset{\text{not}}{=} P^{dr}_p$. La pasul p se calculează $A \leftarrow T_p P^{st}_p A P^{dr}_p$, întregul algoritm fiind:

$$A \leftarrow U = T_{n-1} P^{st}_{n-1} T_{n-2} P^{st}_{n-2} \dots T_1 P^{st}_1 A P^{dr}_1 P^{dr}_2 \dots P^{dr}_{n-1}. \tag{8}$$

Algorithm 4 Eliminarea gaussiană cu pivotare totală

1: **procedure** GPT(A)
2: **for** $p = 1 : n-1$ **do**
3: Determină i_p şi j_p $(p \le i_p, j_p \le n)$ 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: $a_{ip} \leftrightarrow a_{ij_p}$;
9: **end for**
10: **for** $i = p+1 : n$ **do**
11: $a_{ip} \leftarrow \mu_{ip} = a_{ip}/a_{pp}$;
12: **end for**
13: **for** $i = p+1 : n$ **do**
14: **for** $j = p+1 : n$ **do**
15: $a_{ij} \leftarrow a_{ij} - \mu_{ip} a_{pj}$;
16: **end for**
17: **end for**
18: **end for**
19: Return A, v_{st}, v_{dr};
20: **end procedure**

Paşii 12-19 reprezintă algoritmul G pentru matricea permutată. Vectorul v_{st} memorează permutările de linii iar v_{dr} memorează permutările de coloane. Complexitatea algoritmului GPP este aceeaşi cu a lui G.

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 menţiunea că $a_1 = 0$ şi $c_n = 0$, iar componentele x_0 şi x_{n+1} nu sunt definite. Forma generală este:

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix} \tag{9}$$

Algoritmul presupune modificarea coeficienţilor sistemului şi aducerea lor la o formă prin care sistemul de poate rezolva direct prin substituţie înapoi.

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & i = 1 \\ \dfrac{c_i}{b_i - c'_{i-1} a_i} & i = 2 : n-1 \end{cases} \tag{10}$$

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & i = 1 \\ \dfrac{d_i - d'_{i-1}a_i}{b_i - c'_{i-1}a_i} & i = 2 : n. \end{cases} \tag{11}$$

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

$$\begin{cases} x_n = d'_n \\ x_i = d'_i - c'_i x_{i+1} \quad i = n-1 : 1. \end{cases} \tag{12}$$

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

Algoritmul lui Thomas este utilizat deoarece este rapid şi apare frecvent în practică: interpolări, ecuaţii diferenţiale, etc. Deşi situaţia este rară, algoritmul poate fi instabil dacă $b_i - c'_{i-1}a_i = 0$ sau numeric zero pentru orice i. Aceasta se întâmplă dacă matriciea este singulară, dar în cazuri rare se poate întampla şi pentru matrici nesingulare. Condiţia de stabilitate este, $\forall i$:

$$|b_i| > |a_i| + |c_i| \tag{13}$$

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

Implementarea algoritmului Thomas este dată mai jos:

```
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;
  end
  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);
end
```

Listing 1: Implementarea în Matlab pentru algoritmul Thomas.

Problema 1

Implementaţi în MATLAB algoritmii G, GPP, GPPS şi GPT.

Problema 2

Modificaţi algoritmii GPP şi GPT pentru a lucra cu matrici superior Hessemberg. Imoplementaţi H_GPP şi H_GPT.

Problema 3

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

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

Scrieţi o funcţie MATLAB cu semnătura `function x = solve(a,b,c,d)` care rezolvă sistemul de ecuaţii $Ax = d$.