Transformări ortogonale: Householder şi Givens. Algoritmul Gram-Schmidt. Polinoame ortogonale

Noţiuni teoretice

Vectori ortogonali. Matrice ortogonală

Fie vectorii coloană $x, y \in R^n$ de forma
$x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ şi $y = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}^T$. Definim:

- *produsul scalar*: $< x, y >= \sum_{i=1}^{n} x_i y_i = y^T x$;
- *norma euclidiană*: $||x||_2 = \sqrt{< x, x >} = \sqrt{x^T x} = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$.

Spunem că:

- vectorii x şi y sunt *ortogonali* dacă $< x, y > = 0$;
- vectorii x şi y sunt *ortonormaţi* dacă sunt ortogonali, $||x||_2 = 1$ şi $||y||_2 = 1$.

O matrice $H \in R^{n \times n}$ este *ortogonală* dacă are coloanele vectori ortonormaţi. Mai mult, o matrice ortogonală are proprietăţile următoare:

1. $H^T H = H H^T = I_n$;
2. $H^{-1} = H^T$;
3. $||Hx||_2 = ||x||_2$;
4. $||H||_2 = 1$;

5. $||HA||_2 = ||A||_2$;
6. $det(H) = \pm 1$.

Transformarea Householder

Metoda propusă iniţial de Alston Scott Householder este folosită pentru a transforma o matrice $A \in R^{n\times n}$ simetrică într-o matrice tridiagonală cu aceleaşi valori proprii. Prezentarea generală a metodei poate fi gasită la adresa [1].

Pentru o matrice $A \in R^{m\times n}$, definim transformarea Householder $R = HA$ folosind reflectori elementari Householder H_p, astfel încât:

$$H = H_{min(m-1,n)} \dots H_p \dots H_2 H_1, \quad A = H^T R.$$

Un reflector elementar Householder H_p este dat de relaţia:

$$H_p = I_m - 2\frac{v_p v_p^T}{v_p^T v_p}$$

unde:

- $v_p = \begin{bmatrix} 0 & 0 & \cdots & v_{pp} & \cdots & v_{mp} \end{bmatrix}^T$ se numeşte vector Householder;
- $v_{pp} = a_{pp} + \sigma_p$;
- $v_{ip} = a_{ip}, \forall i > p$;
- $a_p = \begin{bmatrix} a_{1p} & a_{2p} & \cdots & a_{pp} & \cdots & a_{mp} \end{bmatrix}^T$ este coloana p din matricea A;
- $\sigma_p = sign(a_{pp})\sqrt{\sum_{i=p}^{m} a_{ip}^2}$;
- $\beta_p = \sigma_p v_{pp}$.

Folosind un reflector elementar Householder putem aduce o matrice la forma superior triunghiulară astfel:

Datorită formei reflectorilor elementari Householder, înmulţirea $A = H_p A$ se efectuează astfel:

- Coloanele $1 : p-1$ din matricea A rămân neschimbate;
- Coloana p din matricea A se modifică astfel:
  - elementele de pe liniile $1 : p-1$ rămân neschimbate;
  - $a_{pp} = -\sigma_p$;

[1] http://mathfaculty.fullerton.edu/mathews/n2003/HouseholderMod.html

Algorithm 1 Transformarea unei matrice la forma superior triunghiulară

procedure [Q, R] = ST(A)
    [m,n] = size(A);
    H = I_n;
    for $p = 1 : min(m-1, n)$ **do**
        $H_p = I_n - 2\frac{v_p v_p^T}{v_p^T v_p}$;
        A = H_p A;
        H = H_p H;
    end for
    Q = H^T;
    R = A;
end procedure

  - $a_{ip} = 0, \forall i > p$;
- Coloanele $j = p + 1 : n$ din matricea A se modifică astfel:
  - elementele de pe liniile $1 : p - 1$ rămân neschimbate;
  - $a_{ij} = a_{ij} - \tau_j \cdot v_{ip}$, $\forall i \geq p$, $\tau_j = \frac{\sum_{i=p}^{m} v_{ip} \cdot a_{ij}}{\beta_p}$.

Transformarea Givens

Metoda Givens este folosită pentru a descompune o matrice $A \in R^{m \times n}$ astfel:

$$A = G^T R$$

unde:

$$G = G_{n-1,m} G_{n-2,m} G_{n-2,m-1} \ldots G_{1n} \ldots G_{13} G_{12}, \quad R = GA.$$

O matrice de rotaţie Givens, notată G_{kl} este folosită pentru a elimina (a anula) elementul $A(l, k)$ de sub diagonala principală ($k < l$) şi are forma:

$$G_{kl} = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cos\theta & \cdots & -\sin\theta & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Pentru a determina matricea de rotaţie Givens G_{kl}, vom folosi relaţiile:

$$\rho = \sqrt{A(k,k)^2 + A(l,k)^2}$$

$$s = \sin\theta = -\frac{A(l,k)}{\rho}$$

$$c = \cos\theta = \frac{A(k,k)}{\rho}$$

Datorită formei matricelor de rotaţie Givens, înmulţirea $G_{kl}x$, unde x este vector coloană, se efectuează astfel:

- $x(k) = c \cdot x(k) - s \cdot x(l)$;
- $x(l) = s \cdot x(k) + c \cdot x(l)$;
- restul elementelor rămân neschimbate.

Algoritmul Gram-Schmidt

Vom considera relaţia $A = QR$, adică:

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}$$

cu necunoscutele q_i şi $r_{ij}, (i <= j)$, unde:

- a_i este coloana i din matricea A;
- q_i este coloana i din matricea Q;
- $r_{ij}, i \leq j$, reprezintă elementele din matricea superior triunghiulară R.

Algoritmul Gram-Schmidt este schiţat în continuare:

Pentru $j = 1 : n$

$r_{ij} = q_i^T a_j, \quad i = 1 : j-1;$

$aux = a_j - \sum_{i=1}^{j-1} r_{ij} q_i;$

$r_{jj} = ||aux||_2;$

$q_j = \frac{aux}{r_{jj}}.$

Algoritmul Gram-Schmidt modificat

Algoritmul Gram-Schmidt clasic prezintă o stabilitate numerică slabă. Acest algoritm poate fi îmbunătăţit folosind următoarea variantă:

Pentru $i = 1 : n$

$r_{ii} = ||a_i||_2;$

$q_i = \frac{a_i}{r_{ii}};$

Pentru $j = i + 1 : n$

$r_{ij} = q_i^T a_j;$

$a_j = a_j - q_i r_{ij}.$

Polinoame ortogonale

Un polinom ortogonal este definit prin:

- relaţia de recurenţă;
- cazurile de bază ale relaţiei de recurenţă;
- intervalul [/(a, b)/] pe care este definit;
- funcţia pondere $w(x)$, folosită la calcularea produsului scalar.

Exemple de polinoame ortogonale:

- *Cebâşev*:

$$T_{n+1} - 2xT_n + T_{n-1} = 0, \quad T_0 = 1, \quad T_1 = x;$$

$$(-1, 1); \quad w(x) = \frac{1}{\sqrt{1 - x^2}};$$

- *Legendre*:

$$(n+1)L_{n+1} - (2n+1)xL_n + nL_{n-1} = 0, \quad L_0 = 1, \quad L_1 = x;$$

$$[-1, 1]; \quad w(x) = 1;$$

- *Laguerre*:

$$G_{n+1} - (2n+1-x)G_n + n^2 G_{n-1} = 0, \quad G_0 = 1, \quad G_1 = 1 - x;$$

$$[0, \infty); \quad w(x) = e^{-x};$$

- *Hermite*:

$$H_{n+1} - 2xH_n + 2nH_{n-1} = 0, \quad H_0 = 1, \quad H_1 = 2x;$$

$$(-\infty, \infty); \quad w(x) = e^{-x^2}$$

.

Proprietăţile polinoamelor ortogonale sunt:

1. Orice polinom ortogonal are radacinile în intervalul [/(a, b)/], reale şi distincte;
2. Orice polinom ortogonal este ortogonal cu orice polinom de grad mai mic decât el.

Polinoamele $p_0, p_1, \ldots, p_n$ reprezintă o bază de polinoame ortogonale, dacă:

- $||p_i|| = 1, \quad \forall i$; și
- $< p_i, p_j >= 0, \quad \forall i \neq j$, unde $< p_i, p_j >= \int_a^b p_i(x)p_j(x)w(x)dx$.

Operații utile cu polinoame în Octave:

```
function test_poly
    %%% Coeficientii polinomului p
    p = [ 2 1 -1]
    %%% Afisarea polinomului
    polyout(p, 'x')
    %%% Coeficientii polinomului q
    q = [ 1 2]
    %%% Afisarea polinomului
    polyout(q, 'x')
    %%% Produsul dintre p si q
    r = conv(p,q)
```

```
    %%% Afisarea rezultatului
    polyout(r, 'x')
endfunction
```

Listing 1: Polinoame în Octave.

Probleme rezolvate

Problema 1

Dacă H_1, H_2 sunt matrici ortogonale, arătaţi că produsul $H_1 H_2$ este o matrice ortogonală.

Soluţie:

Din proprietăţile matricelor ortogonale, avem: H ortogonală $\Rightarrow HH^T = I_n$ și

$$\begin{cases} H_1 H_1^T = I_n \\ H_2 H_2^T = I_n \end{cases} \Rightarrow (H_1 H_2)(H_1 H_2)^T = H_1 H_2 H_2^T H_1^T = I_n$$

Problema 2

Fie un reflector elementar Householder

$$H = I_n - \frac{2uu^T}{\|u\|^2}$$

.

a) Arătaţi că $H^T = H$;

b) Calculaţi Hu.

Soluţie:

a) $H^T = \left(I_n - \frac{2uu^T}{\|u\|^2}\right)^T = I_n - \frac{2uu^T}{\|u\|^2} = H$

b) $Hu = \left(I_n - \frac{2uu^T}{\|u\|^2}\right) u = u - \frac{2uu^T}{\|u\|^2} u = u - 2u = -u$

Problema 3

Determinaţi factorizarea QR folosind transformarea Householder pentru matricea:

$$A = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix}.$$

Soluţie:

Prima iteraţie:

$$a_1 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \quad \sigma_1 = \|a_1\|_2 = 3 \quad u_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix} \quad \|u_1\|_2^2 = 6;$$

$$H_1 = I_3 - \frac{2u_1u_1^T}{\|u_1\|_2^2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 1 & -1 & -2 \\ -1 & 1 & 2 \\ -2 & 2 & 4 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix}$$

$$A_2 = H_1A_1 = \frac{1}{3}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & -2 \\ 2 & -2 & -1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 1 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 0 & 3 \\ 0 & 3 & 3 \end{bmatrix}$$

A doua iteraţie:

$$\overline{a_2} = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \quad \sigma_2 = \|\overline{a_2}\|_2 = 3 \quad u_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \quad \|u_2\|_2^2 = 2;$$

$$H_2 = I_3 - \frac{2u_2u_2^T}{\|u_2\|_2^2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$A_3 = H_2A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 3 & 3 & 3 \\ 0 & 0 & 3 \\ 0 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

Atunci:

$$R = A_3 = \begin{bmatrix} 3 & 3 & 3 \\ 0 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

$$Q = (H_2H_1)^T = H_1H_2 = \frac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ 1 & -2 & 2 \\ 2 & -1 & -2 \end{bmatrix}$$

Problema 4

Să se scrie un program OCTAVE care să implementeze transformarea Givens. Programul primeşte ca date de intrare: A - matricea sistemului; b - vectorul termenilor liberi. Rezultatele programului vor fi: Q - matricea factor ortogonală; R - matricea factor superior triunghiular; b - vectorul modificat al termenilor liberi.

```
function [Q R b] = givens(A, b)
  [m n] = size(A);
  Q = eye(m);

  for k = 1 : min(m-1, n)
    for l = k + 1 : m
            r = sqrt(A(k,k)^2 + A(l,k)^2);
            c = A(k,k)/r;
            s = -A(l,k)/r;

            aux = [c -s; s c]*[A(k,k:n); A(l,k:n)];
            A(k,k:n) = aux(1, :);
            A(l,k:n) = aux(2, :);

            aux = [c -s; s c]*[b(k); b(l)];
            b(k) = aux(1);
            b(l) = aux(2);

            aux = [c -s; s c]*[Q(k,1:m); Q(l,1:m)];
            Q(k,1:m) = aux(1, :);
            Q(l,1:m) = aux(2, :);
      end
    end
```

```

  Q = Q';
  R = A;
end
```

Listing 2: Transformarea Givens.

Problema 5

Să se scrie un program OCTAVE care să implementeze algoritmul Gram-Schmidt. Programul primeşte ca date de intrare: A - matrice. Rezultatele programului vor fi: Q - matricea factor ortogonală; R - matricea factor superior triunghiular.

Soluţie:

```
function [Q, R] = Gram_Schmidt(A)
  [m n] = size(A);
  Q = zeros(m,n);
  R = zeros(n);

  for j = 1 : n
    for i = 1 : j-1
      R(i,j) = Q(:,i)' * A(:,j);
    endfor

    s = zeros(m,1);
    for i = 1 : j-1
      s = s + R(i,j) * Q(:,i);
    endfor
    %%% Echivalent pentru instructiunea for de mai sus:
    %%  s = Q(:, 1:j-1) * R(1:j-1, j);

    aux = A(:,j) - s;

    R(j,j) = norm(aux,2);
    Q(:,j) = aux/R(j,j);
  endfor
endfunction
```

Listing 3: Algoritmul Gram-Schmidt.

Problema 6

Să se scrie un program OCTAVE pentru calculul coeficienţilor unui polinom ortogonal de grad n. Selecţia polinomului se face printr-un parametru şir de caractere care poate avea valorile: '*cebasev*', '*legendre*', '*laguerre*' sau '*hermite*'.

Soluţie:

```
function P = poliOrtogonal(nume_polinom, n)

  if strcmp(nume_polinom, 'legendre')
    P0 = [1];
    P1 = [1 0];
    C0 = @(n)([-n/(n+1)]);
    C1 = @(n)([(2*n+1)/(n+1) 0]);
  endif

  if strcmp(nume_polinom,  'cebasev')
    P0 = [1];
    P1 = [1 0];
    C0 = @(n)([-1]);
    C1 = @(n)([2 0]);
  endif

  if strcmp(nume_polinom, 'laguerre')
    P0 = [1];
    P1 = [-1 1];
    C0 = @(n)([-n^2]);
    C1 = @(n)([-1 2*n+1]);
  endif

  if strcmp(nume_polinom,  'hermite')
    P0 = [1];
    P1 = [2 0];
    C0 = @(n)([-2*n]);
    C1 = @(n)([2 0]);
  endif

  P = poliOrtogonalGeneral(n, P0, P1, C0, C1);
endfunction
```

```
function P = poliOrtogonalGeneral(n, P0, P1, C1, C2)
  %Pn+1 = C1(n)*Pn + C2(n)*Pn-1
  %Valori initiale: P0, P1
  %Iesire: Pn

  %Cazuri de baza
  if n == 0
    P = P0;
  endif

  if n == 1
    P = P1;
  endif

  %Ultimii 2 termeni ai recurentei sunt pastrati in P0 si P1
  for i=2:n
    Paux1 = conv(P0, C1(i));
    Paux2 = conv(P1, C2(i));

    m = length(Paux1);
    n = length(Paux2);
    r = max(m, n);

    Paux1(r-m+1:r) = Paux1(1:m);
    Paux1(1:r-m) = zeros(1,r-m);

    Paux2(r-n+1:r) = Paux2(1:n);
    Paux2(1:r-n) = zeros(1,r-n);

    Paux = Paux1 + Paux2;

    P0 = P1;
    P1 = Paux;
  endfor

  P = P1;
endfunction
```

Probleme propuse

Problema 1

Se consideră vectorii $u, v \in R^n$ ortonormaţi ($||u||_2 = 1$, $||v||_2 = 1$, $u^T v = v^T u = 0$). Se formează vectorul $x = u + v$.

a) Să se dea exemplu de doi vectori ortonormaţi;

b) Să se calculeze $||x||_2$;

c) Se formează matricea $H = I_n - xx^T$. Să se calculeze Hu, Hv şi $||H||_2$;

d) Dacă $A = uv^T$, calculaţi $B = H^{-n}AH^n$.

Problema 2

Implementaţi transformarea Householder pentru o matrice A, astfel:

a) Implementaţi o funcţie care primeşte un vector x, un index p şi calculează parametrii σ, v_p, β definiţi mai sus:

```
function [vp, sigma, beta] = GetHSReflector(x, p)
```

b) Implementaţi o funcţie care primeşte un vector x, un index p, parametrul $sigma$ şi calculează transformarea Householder aplicată asupra vectorului, considerând că acest vector a fost folosit la calculul parametrilor (coloana p din A).

```
function x = ApplyHSToPColumn(x, p, sigma)
```

c) Implementaţi o funcţie care primeşte un vector oarecare x, un index p, vectorul Householder v_p parametrul $beta$ şi calculează transformarea Householder aplicată asupra vectorului (coloanele $p + 1 : n$ din A).

```
function x = ApplyHSToRandomColumn(x, vp, p, beta)
```

d) Implementaţi funcţia

```
function [Q, R] = Householder(A),
```

folosind funcţiile definite mai sus.

Problema 3

Să se determine descompunerea QR pentru matricea $A = \begin{bmatrix} 3 & 1 & -2 \\ 1 & 3 & 1 \\ -2 & 1 & 3 \end{bmatrix}$ folosind transformarea Givens.

Problema 4

Să se scrie un program OCTAVE pentru a implementa algoritmul Gram-Schmidt modificat.