Curs 4.

Metode de rezolvare a sistemelor liniare bazate pe factorizare ortogonală.

Sistemul supradeterminat de ecuaţii liniare

$Ax=b, \quad A\in R^{mxn}, \quad b \in R^{m}, \quad m>n$

nu admite în general soluţie.

Soluţia în sensul celor mai mici pătrate (sau *pseudosoluţia*) se defineşte ca vectorul x^* din R^n care asigură minimizarea normei euclidiene a *vectorului reziduu:*

$$r(x^*) = \|b - A \cdot x^*\|_2 = \min_{x\in R^n} r(x) = \min_{x\in R^n}\|b - A \cdot x\|_2$$

Pseudosoluţia este soluţie exactă pentru ***sistemul normal***

$A^TAx=A^Tb$

Sistemul normal este rău condiţionat astfel încât metodele obişnuite de rezolvare (Gauss, Cholesky, etc) nu dau rezultate satisfăcătoare, fiind necesare metode mai stabile din punct de vedere numeric, bazate pe triunghiularizare ortogonală.

Pseudosoluţia este unică dacă matricea A are coloanele liniar independente, caz în care se exprimă ca:

$x^*=(A^TA)^{-1}A^Tb=A^\#b$

în care se defineşte

$A^\#=(A^TA)^{-1}A^T\in R^{nxm}$

ca *inversă generalizată* sau *pseudoinversă Penrose - Moore* a matricei dreptunghiulare $A\in R^{mxn}$.

Doi vectori $x, y \in R^n$ sunt ortogonali în raport cu produsul euclidian dacă $x^Ty=0$.

Complementul ortogonal $S^\perp$ a subspaţiului $S \in R^n$ se defineşte ca:

$S^\perp = \{y\in R^n \mid x^Ty=0, x\in S\}$

Vectorii $q_1, q_2, ..., q_n\in R^m$ sunt ortonormaţi, dacă $q_i^Tq_j=\delta_{ij}$

Matricea $Q = [q_1\ q_2\ ...\ q_n] \in R^{mxn}$ este *ortogonală* şi $Q^T.Q=I_n$

Considerăm o matrice ortogonală pătrată $Q \in R^{nxn}$. Matricele ortogonale au următoarele proprietăţi:

$1^0.\ Q^{-1} = Q^T$

$2^0.\ \det(Q) = \pm1$

$\det(Q^T.Q)=\det(Q.Q)=\det^2(Q)=\det(I_n)=1$

În C^n vectorii sunt ortogonali dacă $x^Hy=0$, unde $x^H=\underline{x}^T$ (transpus conjugat) Matricea Q este *unitară* dacă $Q^H.Q=I_n$.

$3^0.\ \|Hx\|_2=\|x\|_2$ –transformarea ortogonală conservă norma euclidiană a unui vector

$\|Hx\|_2^2=(Hx,Hx)=(Hx)^T(Hx)=x^TH^THx=x^T(H^TH)x=x^Tx=\|x\|_2^2$

4^0. $\|H\|_2=1$ -norma matricială euclidiană subordonată este 1:

$$\|H\|_2 = \sup_{\underline{x}\neq\underline{0}} \frac{\|H\cdot\underline{x}\|_2}{\|\underline{x}\|_2} = 1$$

5^0. $cond_2(A) = 1$

6^0. $\|H.A\|_2=\|A\|_2$ -transformarea ortogonală conservă norma euclidiană a unei matrice:

$$R=H.A \Rightarrow \|R\|_2=\|H.A\|_2\leq\|H\|_2.\|A\|_2=\|A\|_2$$

$$A=H^T.R \Rightarrow \|A\|_2=\|H^T.R\|_2\leq\|H\|_2.\|R\|_2=\|R\|_2$$

$$\|H.A\|_2=\|A\|_2$$

O matrice P se numeşte *matrice idempotentă* sau *matrice proiector*, dacă $P^2=P$.

În raport cu un proiector P, un vector $v\in R^n$ poate fi descompus unic în:

$$v = P.v + (I-P).v = v_1 + v_2$$

$$P.v_2 = P.(I-P).v = (P-P^2).v = 0.v = 0$$

$v_1=P.v \in S$ – proiecţia lui v în *spaţiul imagine* a lui P

$v_2=(I-P)v \in S^\perp$ – proiecţia lui v în *spaţiul nul* a lui P

Dacă P este simetric ($P=P^T$), atunci:

$$v_1^Tv_2 =(P.v)^T.(I-P).v=v^T.P.(I-P)= v^T.(P-P^2)v=0 \Rightarrow v_2 \perp S \Rightarrow v_2\in S^\perp$$

P este proiectorul ortogonal în S

$I-P$ este proiectorul ortogonal în $S^\perp$

O *metodă ortogonală* transformă sistemul $Ax=b$ într-un sistem echivalent cu matrice superior triunghiulară $HAx=Hb$ în care matricea de transformare $H\in R^{mxn}$ este ortogonală. Matricea sistemului echivalent (în sensul că are aceeaşi pseudosoluţie cu sistemul iniţial), are acelaşi număr de condiţionare în normă 2. Intr-adevăr, transformările ortogonale conservă norma euclidiană

Metoda Householder

O *matrice ortogonală elementară* se obţine modificând matricea unitate, cu o matrice de rang 1. Reflectorul Householder este o asemenea matrice ortogonală:

$$H = I - \beta uu^T, \qquad \beta = 2 / (u^Tu)$$

H este simetrică:

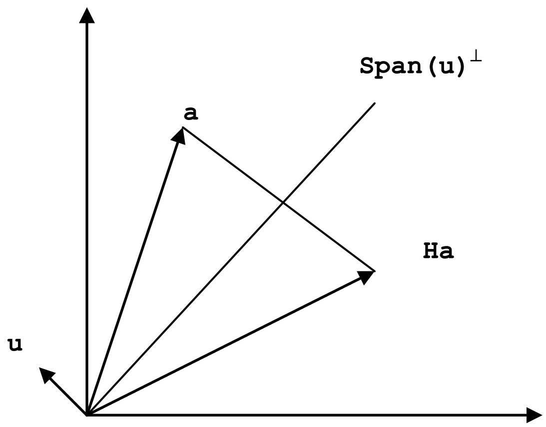
$$H^T=(I - \beta uu^T)^T = I - \beta(uu^T)^T = I - \beta(u^T)^Tu^T = I - \beta uu^T = H$$

H este ortogonală:

$$H^TH = H^2 = I-2\beta uu^T+\beta^2u(u^Tu)u^T = I-2\beta uu^T+\beta^2u(2/\beta)u^T = I$$

u se numeşte *vector Householder*.

$Hu = (I-\beta uu^T)u = u - \beta u(u^Tu) = u - \beta u(2/\beta) = u - 2u = -u$ ceea ce justifică numele de reflector



$$Ha = (I - \beta uu^T)a = a - \beta u(u^Ta)$$

Dacă $a \perp u$ atunci $u^Ta = 0$ şi $Ha = a$

altfel $a - Ha = \beta u(u^Ta)$ adică $a-Ha \parallel u$

Transformarea H reflectă un vector a în raport cu planul normal vectorului u.

Vom determina transformarea H, astfel încât vectorul reflectat Ha să aibă toate componentele 0, cu excepţia primeia dintre ele:

$$Ha = \pm\sigma e_1, \quad \sigma = \|a\|_2$$

$$HHa = \pm\sigma He_1 \quad \Rightarrow \quad He_1 = \pm a/\sigma$$

Vom determina deci matricea ortogonală H, având prima coloană proporţională cu $\pm a/\sigma$

Matricea Householder se foloseşte sub forma echivalentă:

$H=I-2uu^T$, cu $u^Tu=1$ (se observă că în acest caz particular $\beta=2$).

Dacă $x,y\in R^n$, $x\neq y$, $\|x\|_2=\|y\|_2$, $u = \dfrac{x-y}{\|x-y\|_2}$ atunci $(I-2uu^T)x=y$.

$$\|x-y\|_2^2=(x-y)^T(x-y)=x^Tx-x^Ty-y^Tx+y^Ty=2(x^Tx-y^Tx)$$

$$(I-2uu^T)x = x - 2\frac{(x-y)(x-y)^T}{\|x-y\|_2^2}x = x - \frac{2(x-y)(x^Tx-y^Tx)}{2(x^Tx-y^Ty)}$$

$$(I-2uu^T)x = x-(x-y)=y$$

Cu alte cuvinte, dacă doi vectori x şi y au aceeaşi origine şi aceeaşi lungime, atunci putem determina o transformare (rotaţie), care să aducă pe unul din ei peste celălalt.

Suntem interesaţi ca unul din vectori să fie parallel cu o axă de coordinate, adică dacă notăm lungimea $\sigma=\pm\|x\|_2$, şi alegem $v=x+\sigma e_1$, $u=v/\|v\|_2$ atunci $(I-2uu^T)x=-\sigma e_1$

$$uu^T=vv^T/\|v\|_2^2$$

$$\|v\|_2^2=(x+\sigma e_1)(x+\sigma e_1)=x^Tx+\sigma x^Te_1+\sigma e_1^Tx+\sigma^2e_1^Te_1 = \|x\|_2^2+\sigma^2+2\sigma x_1=2\sigma^2+2\sigma x_1=$$

$$2\sigma(\sigma+x_1)$$

Considerăm $\mathbf{x} \in \mathbf{R}^m$ cu $\sigma^2 = \sum_{i=p}^{m} x_i^2 = \| x(p+1:m) \|_2^2 \neq 0$

Vom aplica vectorului **x** un reflector Householder $\mathbf{H_p}$ pentru a-i anula componentele **x(p+1:m)**

Fixăm componentele vectorul $\mathbf{u_p}$ care intervine în expresia reflectorului $\mathbf{H_p}$ la valorile:

$$u_{ip} = \begin{cases} 0 & i = 1 : p-1 \\ x_p + \sigma & i = p \\ x_i & i = p+1 : m \end{cases}$$

Acest vector poartă numele de *vector Householder*.

$$H_p \cdot x = \left(I_m - \beta \cdot u_p \cdot u_p^T\right) \cdot x = x - \beta\left(u_p^T \cdot x\right) \cdot u_p$$

$$u_p^T \cdot x = \sum_{i=p}^{m} u_{ip} x_i = u_{pp} x_p + \sum_{i=p+1}^{m} u_{ip} x_i = \left(x_p + \sigma\right) \cdot x_p + \sum_{i=p+1}^{m} x_i^2$$

$$u_p^T \cdot x = \left(x_p + \sigma\right) \cdot x_p + \sum_{i=p}^{m} x_i^2 - x_p^2 = \left(x_p + \sigma\right) \cdot x_p + \sigma^2 - x_p^2 = \sigma \cdot \left(x_p + \sigma\right)$$

$$u_p^T \cdot u_p = \|u_p\|_2^2 = \sum_{i=p}^{m} u_{ip}^2 = \left(u_{pp}^2 + \sum_{i=p+1}^{m} x_i^2\right) = \left(x_p + \sigma\right)^2 + \sigma^2 - x_p^2 = 2\sigma\left(x_p + \sigma\right)$$

$$\beta = \frac{2}{u_p^T \cdot u_p} = \frac{1}{\sigma \cdot \left(x_p + \sigma\right)}$$

Pentru a evita anularea numitorului lui **β** vom alege ca **σ**, să aibă acelaşi semn cu $\mathbf{x_p}$

$$\sigma = \mathrm{sign}\left(x_p\right) \cdot \sqrt{\sum_{i=p}^{m} x_i^2} = \mathrm{sign}\left(x_p\right) \cdot \| x(p:m) \|_2$$

$$\left(H_p \cdot x\right)_i = x_i - \beta \cdot \left(u_p^T \cdot x\right) \cdot u_{ip} = x_i - \frac{\sigma \cdot \left(x_p + \sigma\right)}{\sigma \cdot \left(x_p + \sigma\right)} u_{ip} = x_i - u_{ip}$$

$$\left(H_p \cdot x\right)_i = x_i - u_{ip} = \begin{cases} x_i & i = 1 : p-1 \\ x_p - u_{pp} = -\sigma & i = p \\ 0 & i = p+1 : m \end{cases}$$

În urma aplicării transformării $\mathbf{H_p}$, vectorul **x** se modifică astfel:

- primele **p-1** componente rămân nemodificate
- componenta **p** devine *mare* în valoare absolută
- restul componentelor **(p+1:m)** se anulează.

```
function [u, b, x]=HSx(x, p)
% determină reflectorul Housholder care
%   lasa primele p-1 componente neschimbate
%   modifică componenta p la o valoare mare
%   anulează restul componentelor
```

```
x=x(:);    %ne asiguram ca x e vector coloana
m=length(x);
sig=sign(x(p))*norm(x(p:m));
b=1/sig/(x(p)+sig);
u=[zeros(p-1,1); x(p)+sig; x(p+1:m)];
x(p)=-sig;
x(p+1:m)=0;
% H este determinat de v si b
% H=eye(m)-v*v'./b;
```

Particularizăm pentru **p=1**

$$(H_1 \cdot x)_i = \begin{cases} -\sigma, & i = 1 \\ 0, & i = 2:m \end{cases}$$

$$H_1 \cdot x = -\sigma \cdot e_1 = -\| x \|_2 \cdot \mathrm{sign}(x_1) \cdot e_1$$

$$u_{i1} = \begin{cases} x_1 + \sigma \\ x_i, & i = 2:m \end{cases}$$

$$u_1 = x + \sigma \cdot e_1$$

Transformarea H_1 este determinată numai de vectorul **u** şi de β. Aşadar

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_m \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} -\sigma \\ 0 \\ \cdots \\ 0 \end{bmatrix} \quad \text{în care} \quad H_1 = \left\{ \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_m \end{bmatrix}, \beta \right\}$$

Cu excepţia primei componente, vectorii **u** şi **x** coincid, ceea ce ne sugerează să păstrăm pe **u** în **x**, în poziţiile `2:m`, unde apar zerouri.

Cum vectorul **u** este determinat esenţial numai ca direcţie, îl vom norma cu σ: $\bar{u} = \frac{u}{\sigma}$

$$u_{i1} = \begin{cases} \frac{x_1}{\sigma} + 1 = \bar{x}_1 + 1 \\ \frac{x_i}{\sigma} = \bar{x}_i, & i = 2:m \end{cases}$$

Prima componentă u_1 va fi păstrată în β. Într-adevăr, înainte de scalarea cu σ, $\beta = \sigma . u_1$

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_m \end{bmatrix} \xrightarrow{H_1} \begin{bmatrix} -\sigma \\ u_2 \\ \cdots \\ u_m \end{bmatrix}, \quad \beta = u_1$$

```
function [beta, x]=HS1x(x)
% aplica transformarea H1 unui vector x
% intoarce beta=u(1) si u(2:m) in x
  x=x(:);
  m=length(x);
```

```
  beta=0;
  sig=norm(x);
  if sig~=0
    if x(1)~=0
      sig = sig*sign(x(1));
    end;
    x(1:m)=x(1:m)/sig;
    beta= 1+x(1);
    x(1)=-sig;
  else
    error('elemente deja nule');
  end
```

Întrucât transformarea H_p modifică numai componentele din poziţiile **p:m**, ea se poate realiza aplicând H_1 unui vector **x(p:m)**

Aplicăm reflectorul H_1 unui vector oarecare **y**

$$H_1 \cdot y = \left(I_m - \beta \cdot u_1 \cdot u_1^T\right) \cdot y = y - \beta \cdot \left(u_1^T \cdot y\right) \cdot u_1 = y - \rho \cdot u_1$$

$$\rho = \beta \cdot u_1^T \cdot y = \beta \cdot \sum_{i=1}^{m} u_{i1} \cdot y_i$$

$$\left(H_1 \cdot y\right)_i = y_i - \rho \cdot u_{i1},\ i = 1 : m$$

```
function y=HS1y(x, beta, y)
% aplica reflectorul H1 unui vector oarecare y
    if beta ~= 0
      t = x(1);
      x(1) = beta;
      m=length(x);
      ro=v(1:m)'*y(1:m)*beta;
      y(1:m)=y(1:m)-ro.*x(1:m);
    end
```

În locul rezolvării sistemului **A.x=b**, cu $A \in R^{m x n}$, $b \in R^m$ se rezolvă sistemul echivalent **H.A.x=H.b**, cu $H \in R^{m x m}$, **H**-ortogonală, sistem care are aceeaşi condiţionare cu sistemul iniţial (**cond(H.A)=cond(A)**)

Pentru orice matrice $A \in R^{m x n}$ ***există o matrice ortogonală*** $H \in R^{m x m}$ ***astfel încât*** **H.A=R** ***în care*** $R \in R^{m x n}$ ***este o matrice superior triunghiulară***

Matricea ortogonală se formează dintr-un produs

$H = H_n H_{n-1} \ldots H_2 H_1$

$H.A = H_n H_{n-1} \ldots H_2 H_1 A$

$A_{p+1} = H_p A_p$ cu **p=1:n**, pornind cu $A_1 = A$.

De exemplu reflectorul H_1 care anulează prima coloană a matricei $A \in R^{m x n}$ este:

```
[beta, A(:,1)]=HSH1x(A(:,1))
for j=2:n
  A(:,j)=HS1y(A(:,1), beta, A(:,j));
end
```

Matricea are structura $\mathbf{H_p} = \begin{bmatrix} \mathbf{I}_{p-1} & 0 \\ 0 & \mathbf{V}_{n-p+1} \end{bmatrix}$

Spre deosebire de metodele gaussiene, metodele ortogonale asigură elemente diagonale mari în valoare absolută, ceea ce conferă o stabilitate deosebită a metodei.

Vom determina matricea $\mathbf{H_p}$, care transformă matricea $\mathbf{A_p}$, utilizând algoritmul **HSx()**, punând ca vector **x** coloana **p** din $\mathbf{H_p A_p}$

$\mathbf{H_p A_p} = \mathbf{H_p}[\mathbf{a}_1 \ \mathbf{a}_2 \ \ldots \ \mathbf{a}_n] = [\mathbf{H_p a}_1 \ \mathbf{H_p a}_2 \ \ldots \ \mathbf{H_p a}_n]$

Coloanele situate în dreapta coloanei **p, (j>p)** se modifică folosind algoritmul **HSy()** Transformarea $\mathbf{H_p}$ nu modifică coloanele $\mathbf{a_j}$ pentru care $\sigma=0$ (coloanele din stânga coloanei **p**, cu **j<p**).

Acestea au $a_{ij}=0$ pentru **i<p<j.**

```
function A=trorto(A)
% triunghiularizare ortogonala Householder
  [m,n]=size(A);
  H=zeros(m,m);
  for p=1:min(m-1,n)
    % se determina Hp
    sig=sign(A(p,p))*norm(A(p:m,p));
    H(p,p)=A(p,p)+sig;
    H(p+1:m,p)=A(p+1:m,p);
    beta(p)=sig*H(p,p);
    % se aplica Hp coloanelor din A
    for j=p+1:n
      ro = H(p,p:m)*A(p:m,j)/beta(p);
      A(p:m,j)=A(p:m,j)-ro*H(p:m,p);
    end;
    A(p,p)=-sig;
    A(p+1:m,p)=0;
  end
```

Matricea **H** poate fi păstrată în triunghiul inferior din **A** (care se anulează), iar elementele diagonale în vectorul **beta**. În triunghiul superior din **A** vom avea matricea **R**

In rezumat, transformarea aplicată matricei :

- îi lasă neschimbate primele coloane
- modifică elementele din coloana astfel:
    - primele elemente rămân neschimbat
    - elementul diagonal devine mare
    - elementele subdiagonale se anulează
    - modifică elementele din coloanele **j>p** astfel:
        - primele **p-1** elemente rămân neschimbate
        - elementele din poziţiile **p:m** se calculează cu relaţia Householder.

Pornind de la sistemul iniţial, prin aplicarea transformărilor:

$\mathbf{A}_{p+1} = \mathbf{H_p A_p}, \qquad \mathbf{A}_1=\mathbf{A}, \qquad p=1:n$

$\mathbf{b}_{p+1} = \mathbf{H_p b_p}, \qquad \mathbf{b}_1=\mathbf{b}$

se ajunge la sistemul echivalent superior triunghiular

$A_{n+1}x = b_{n+1}$

Metoda Givens (matrice de rotaţie elementare).

Metoda Givens utilizează matrice elementare de rotaţie, de forma:

$$G_{kl} = \begin{bmatrix} I_{k-1} & & & & \\ & c & & s & \\ & & I_{l-k-1} & & \\ & -s & & c & \\ & & & & I_{m-l} \end{bmatrix}, \qquad c^2+s^2=1,$$

O asemenea matrice defineşte o rotaţie de ordinul **m** în planul **(k,l)** cu unghiul θ, cu $c=\cos\theta$ şi $s=\sin\theta$

Matricea G_{kl} este ortogonală: $G_{kl}*G^T_{kl}=I_n$

Transformarea G_{kl} aplicată unui vector $x\in R^m$ îi modifică numai componentele x_k şi x_l

$G_{kl}x=[x_1 \ldots cx_k+sx_l \ldots -sx_k+cx_l \ldots x_m]'$

```
function x=rotvec(k, l, c, s, x)
%aplica transformarea Gkl vectorului x
  t = zeros(2,1);
  t = [c s;-s c]*[x(k); x(l)];
  x(k) = t(1);
  x(l) = t(2);
```

Determinăm transformarea **(c, s),** care anulează componenta x_l :

$-s.x_k+c.x_l=0$

$$c = \frac{x_k}{\sqrt{x_k^2+x_l^2}} = \frac{x_k}{r} \qquad s = \frac{x_l}{\sqrt{x_k^2+x_l^2}} = \frac{x_l}{r}$$

$$x_k = c\cdot x_k + s\cdot x_l = \frac{x_k^2+x_l^2}{\sqrt{x_k^2+x_l^2}} = \sqrt{x_k^2+x_l^2} = r$$

Rotaţia G_{kl}, care modifică componentele la valorile: $x_k=r$ şi $x_l=0$ este:

```
function [c, s, x]=rot(k, l, x)
% determina rotatia (c,s) care
% anuleaza x(l)
  r=sqrt(x(k)^2+x(l)^2);
  c=x(k)/r;
  s=x(l)/r;
```

În metoda Givens matricea ortogonală **G** se formează ca un produs de *matrice elementare de "rotaţie"* de forma

$$G = \underbrace{G_{n-1,n}}_{G_{n-1}} \cdot \underbrace{G_{n-2,n}\cdot G_{n-2,n-1}}_{G_{n-2}} \cdot \underbrace{G_{1n}\cdot G_{1,n-1}\cdots G_{12}}_{G_1}$$

Matricea G_{kl} afectează prin înmulţire numai liniile **k** şi **l**:

$A^{(p+1)}=G_{kl}A^{(p)}$,   k=1:n-1,   l+k+1:n, $A^{(0)}=A$

Plecând cu matricea **A** şi folosind transformările ortogonale G_{kl} se obţine o matrice superior triunghiulară:

```
function A=trrot(A)
% triunghiularizare ortogonala cu rotatii
  [m,n]=size(A);
  t=zeros(2,n);
  for k=1:min(m-1,n)
    for l=k+1:m
      [c,s]=rot(k,l,A(:,k));
      t=[c s;-s c]* [A(k,1:n);A(l,1:n)];
      A(k,1:n)=t(1,1:n);
      A(l,1:n)=t(2,1:n);
    end
 end
```

Factorizarea QR

Orice matrice $A\in R^{mxn}$ *cu coloane independente, poate fi descompusă sub forma* **A=QR**, *cunoscută sub numele de factorizare* **QR** *în care* **Q**= *este o matrice cu coloane ortogonale şi* **R** *este o matrice pătrată superior triunghiulară.*

Pornim de la teorema Householder: $\forall A\in R^{mxn}$, **A**-matrice cu coloane liniar independente, $\exists H\in R^{mxm}$, **H** – ortogonală ($H^TH=I_m$) astfel încât $H.A = R$, cu $R\in R^{mxn}$, **R** – superior triunghiulară.

$H^T.H.A = A = H^T.R = Q.R$

Factorizarea **QR** admite şi o “reprezentare economică”, obţinută prin partiţionarea matricei **Q**:

$$Q = \underset{\begin{array}{cc} n & m-n \end{array}}{\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}} \cdot R = \begin{array}{c} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \\ n \end{array} \begin{array}{c} n \\ m-n \end{array}$$

$Q_1 \in R^{m\times n}, Q_2 \in R^{(m-n)\times n}, R_1 \in R^{n\times n}$.

$Q^T=H=H_nH_{n-1}...H_2H_1I_m$

$$Q_1 = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \cdot \begin{bmatrix} I_n \\ 0 \end{bmatrix} = Q \cdot \begin{bmatrix} I_n \\ 0 \end{bmatrix} = H_1H_2 \dots H_n \cdot \begin{bmatrix} I_n \\ 0 \end{bmatrix}$$

```
[x, Q, R] = Householder(A, b)
%Intrări: A = matricea sistemului
%         b = vectorul termenilor liberi
%Ieşiri:  x = vectorul necunoscutelor
%  Q = matricea ortogonală
%  R = matricea superior triunghiulară
   Q = eye(m);
```

```
   for k = 1:n
     s = norm(A(k:m,k)).^2;
     if A(k,k) < 0
       s = -s
     end;
     v(1:k-1) = 0;
     v(k) = A(k,k) + s;
     A(k,k) = -s;
     v(k+1:m) = A(k+1:m,k)
     A(k+1:m,k) = 0
     p = s * v(k);
% Ak+1 = Qk * Ak
     for j = k+1:n
          t = v(k:m)' * A(k:m,j) / p;
          A(k:m,j) = A(k:m,j) - t * v(k:m);
     end;
     % bk+1 = Qk * bk
     t = v(k:m)' * b(k:m) / p;
     b(k:m) = b(k:m) - t * v(k:m);
     % Q=Qk * Q
     for j = k+1:n
        t = v(k:m)' * Q(k:m,j) / p;
        Q(k:m,j) = Q(k:m,j) - t * v(k:m);
     end;
   end;
% rezolvare sistem triunghiular
   for i = n:-1:1
       s = A(i,i+1:n) * x(i+1:n);
       x(i) =(b(i) - s) / A(i,i);
   end;
   R = triu(A);
   Q = Q';

[x, Q, R] = Givens(A, b)
 % Intrări: A = matricea sistemului
 %          b = vectorul termenilor liberi
 % Ieşiri:  x = vectorul necunoscutelor
 %      Q = matricea factor ortogonală
 %      R = factorul superior triunghiular
 Q=eye(m);
   t=zeros(2,n);
   u=zeros(2,1);
   for k = 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;
       t  =  [c s; -s c]* [A(k,1:n); A(l,1:n)];
       A(k,1:n) = t(1);
       A(l,1:n) = t(2);
       u    = [c s;-s c]*[b(k); b(l)];
       b(k) = t(1);
```

```
      b(l) = t(2);
      t = [c s;-s c] * [Q(k,1:m);  Q(l,1:m)];
      Q(k,1:m)= t(1);
      Q(l,1:m) =t(2);
    end
  end;
  for i = n:-1:1
      s    = A(i,i+1:n) * x(i+1:n);
      x(i) = (b(i) - s) / A(i,i);
  end
  R = A; Q = Q';
```

În MATLAB apelul `[Q,R] = qr(A)` produce descompunerea matricei **A** într-o matrice superior triunghiulară $\mathbf{R}\in\mathbf{R}^{m\times n}$ (de aceeaşi dimensiune cu **A**) şi o matrice $\mathbf{Q}\in\mathbf{R}^{m\times m}$ unitară astfel încât **A=Q*R.**

`[Q,R] = qr(A,0)` produce o descompunere "economică", în care se calculează numai primele **n** coloane ale matricei **Q**.

Ortogonalizarea Gram-Schmidt

$$\begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_j & \cdots & \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_j & \cdots & \mathbf{q}_n \end{bmatrix} \cdot \begin{bmatrix} r_{11} & \cdots & r_{1j} & \cdots & r_{1n} \\ 0 & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & r_{jj} & \cdots & r_{jn} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & r_{nn} \end{bmatrix}$$

Coloanele matricei **Q** sunt ortogonale (formează o bază ortonormată):

$$q_i^T \cdot q_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

$$a_1 = q_1 \cdot r_{11}$$

$\| q_1 \| = 1$ de unde $r_1 = \|a_1\|$ şi $q_1 = a_1 / r_{11}$

$a_2=q_1r_{12}+q_2r_{22}$, $\qquad q_1^Ta_2=r_{12}$,

$\|q_2\|=1$, $\qquad r_{22}=\|a_2-q_1r_{12}\|$, $q_2=(a_2-q_1r_{12})/r_{22}$

$a_3=q_1r_{13}+q_2r_{23}+q_3r_{33}$, $\qquad q_2^Ta_3=r_{23}$,

$\|q_3\|=1$, $\qquad r_{33}=\|a_3-q_1r_{13}-q_2r_{23}\|$, $q_3=(a_3-q_1r_{13}-q_2r_{23})/r_{33}$

$a_j=q_1r_{1j}+q_2r_{2j}+\ldots+q_jr_{jj}$ $\qquad q_k^Ta_j=r_{kj}$, $k=1:j-1$

$\|q_j\|=1$, $\qquad r_{jj} = \left\| a_j - \sum_{k=1}^{j-1} q_k \cdot r_{kj} \right\| \qquad q_j = \left(a_j - \sum_{k=1}^{j-1} q_k \cdot r_{kj} \right) / r_{jj}$

pentru $j=1:n$
  $q_j = a_j$
  pentru $k=1:j-1$
    $r_{kj} = q_k{}' * q_j$
    $q_j = q_j - \sum_{k=1}^{j-1} q_k r_{kj}$

$r_{jj}=\|q_j\|$
$q_j = q_j/r_{jj}$

Algoritmul Gram-Schmidt clasic este numeric instabil. Rezultate mult mai bune ne furnizează o variantă modificată

$q_1' a_k=r_{1k}$
$(I-q_1q_1') a_k=q_1r_{2k}+...+q_kr_{kk}$
$q_2q_2'(I-q_1q_1') a_k=q_2r_{2k}$

$q_1q_1' a_k=q_1r_{1k}$
$q_2'(I-q_1q_1') a_k=r_{2k}$
$(I-q_1q_1')(I-q_2q_2') a_k=q_3r_{3k}+...+q_kr_{kk}$

```
Q = A
pentru k=1:n
  rkk=‖qk‖
  qk=qk/rkk
  pentru j=k+1:n
     rkj=qk'qj
     qj=qj-qkrkj
```

Algoritmul Gram-Schmidt clasic

```
[Q, R] = Gram_Schmidt(m, n, A)
% Intrări:
A = matricea de factorizat (baza iniţială)
% Ieşiri:Q = factorul ortogonal(baza
% ortonormată)
% R = factorul superior triunghiular
   [m,n]=size(A);
   for i = 1 : n
     R(1:i-1,i) = Q(1:m,1:i-1)'*A(1:m,i);
     y = A(1:m,i)-Q(1:m,1:i-1)*R(1:i-1,i);
     R(i,i) = norm(y);
     Q(1:m,i) = y ./ R(i,i);
   end
```

Algoritmul Gram-Schmidt modificat

```
[Q, R] = Gram_Schmidt_Modificat(m, n, A)
%Intrări:  A = matricea de factorizat
% (baza iniţială)
%Ieşiri:Q =  baza ortonormată
%       R = factorul superior triunghiular
   [m,n]=size(A);
  for i = 1 : n
    R(i,i) =norm(A(1:m,i));
    Q(1:m,i) = A(1:m,i) / R(i,i);
    for j = i+1 : n
      R(i,j) = Q(1:m,i)' *A(1:m,j);
      A(1:m,j)= A(1:m,j)- Q(1:m,i)*R(i,j);
    end
  end
```