Calcul Numeric

Curs 6

Conf. dr. Anca Ignat

Conf. dr. Andreea Arusoaie

March 22, 2026

Descompuneri QR - Definiție

Fie $A \in \mathbb{R}^{n \times n}$ o matrice reală pătratică.
Presupunem că avem:

$$A = QR$$

unde Q este o matrice ortogonală iar R este o matrice superior triunghiulară.

Atunci:

$$A\mathbf{x} = \mathbf{b} \Leftrightarrow QR\mathbf{x} = \mathbf{b} \Leftrightarrow Q^T QR\mathbf{x} = Q^T\mathbf{b} \Leftrightarrow R\mathbf{x} = Q^T\mathbf{b}$$

.

Descompuneri QR-Algoritmul lui Householder

O **matrice de reflexie** este o matrice $P \in \mathbb{R}^{n\times n}$ de forma:

$$P = I_n - 2\mathbf{v}\mathbf{v}^T, \qquad \mathbf{v} \in \mathbb{R}^n,\ \mathbf{v} \neq 0$$

cu

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{j=1}^{n} |v_j|^2} = 1.$$

Astfel, avem

$$\mathbf{v}\mathbf{v}^T = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} (v_1 v_2 \dots v_n)^T = \begin{pmatrix} v_1^2 & v_1 v_2 & \cdots & v_1 v_n \\ v_2 v_1 & v_2^2 & \cdots & v_2 v_n \\ \vdots & & & \\ v_n v_1 & v_n v_2 & \cdots & v_n^2 \end{pmatrix}$$

Algoritmul lui Householder

Matricele de reflexie sunt:

- **simetrice**: $P^T = P$.

$$P^T = (I_n - 2\mathbf{v}\mathbf{v}^T)^T = I_n - 2(\mathbf{v}\mathbf{v}^T)^T = I_n - 2(\mathbf{v}^T)^T\mathbf{v}^T = I_n - 2\mathbf{v}\mathbf{v}^T = P$$

- **ortogonale**:

$$P^T P = PP^T = I_n$$

Într-adevăr,

$$\begin{aligned} P^2 &= (I_n - 2\mathbf{v}\mathbf{v}^T)(I_n - 2\mathbf{v}\mathbf{v}^T) = I_n - 2\mathbf{v}\mathbf{v}^T - 2\mathbf{v}\mathbf{v}^T + 4(\mathbf{v}\mathbf{v}^T)(\mathbf{v}\mathbf{v}^T) \\ &= I_n - 4\mathbf{v}\mathbf{v}^T + 4\mathbf{v}(\mathbf{v}^T\mathbf{v})\mathbf{v}^T = I_n - 4\mathbf{v}\mathbf{v}^T + 4\mathbf{v}\|\mathbf{v}\|_2^2\mathbf{v}^T \\ &= I_n - 4\mathbf{v}\mathbf{v}^T + 4\mathbf{v}\mathbf{v}^T = I_n \quad (\|\mathbf{v}\|_2 = 1) \end{aligned} \tag{1}$$

Algoritmul lui Householder

<u>*Exemplu*</u>: $n = 2,\ \mathbf{v} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad P = I_2 - 2\mathbf{v}\mathbf{v}^T.$

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Fie $\mathbf{x} \in \mathbb{R}^2,\ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ şi fie $\mathbf{y} = P\mathbf{x}$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -x_1 \\ x_2 \end{pmatrix}$$

Vectorul $\mathbf{y} = P\mathbf{x}$ este reflectatul vectorului $\mathbf{x}$ în raport cu axa Ox_2.

Algoritmul lui Householder

Algoritmul care folosește matricele de reflexie pentru a obține o descompunere QR pentru o matrice $A \in \mathbb{R}^{n \times n}$ a fost descris de Alston S. Householder în articolul *"Unitary triangularization of a nonsymmetric matrix"* apărut în *Journal of the Assoc. of Computing Machinery* **5** (1958), 339–342.

Transformarea matricei A într-una superior triunghiulară se face în $(n-1)$ pași, la fiecare pas folosindu-se o matrice de reflexie.

Algoritmul Householder

Transformarea matricei A într-una superior triunghiulară se face în $(n-1)$ pași, la fiecare pas folosindu-se o matrice de reflexie.

Pas 1:

$$A^{(1)} = P_1 A$$

- matricea P_1 se alege astfel încât coloana 1 să fie transformată în formă superior triunghiulară.

Pas 2:

$$A^{(2)} = P_2 A^{(1)} = P_2(P_1 A)$$

- P_2 transformă coloana 2 în formă superior triunghiulară, fără să schimbe coloana 1.

Algoritmul Householder

Pas r:

$$A^{(r)} = P_r A^{(r-1)} = P_r(P_{r-1} \cdots P_1 A)$$

- se transformă coloana r în formă superior triunghiulară fără să schimbe primele $(r-1)$ coloane.

$\vdots$

Pas $n-1$:

$$A^{(n-1)} = P_{n-1} A^{(n-2)} = P_{n-1}(P_{n-2} \cdots P_1 A) = R$$

- se transformă coloana $n-1$ a matricei $A^{(n-2)}$ în formă superior triunghiulară fără să schimbe primele $(n-2)$ coloane.

  $A^{(n-1)}$ este în formă superior triunghiulară.

Algoritmul Householder

Descompunerea QR construită cu algoritmul lui Householder este următoarea:

$$P_{n-1}P_{n-2} \cdot \ldots \cdot P_r \cdot \ldots P_2P_1A = \tilde{Q}A = R$$

unde

$$\tilde{Q} = P_{n-1} \cdot \ldots \cdot P_r \cdot \ldots \cdot P_2P_1$$

Q este matrice ortogonală ca produs de matrice ortogonale.

$$\tilde{Q}A = R \;\Leftrightarrow\; \tilde{Q}^TQA = \tilde{Q}^TR \;\Leftrightarrow\; A = \tilde{Q}^TR = QR \;\Leftrightarrow\; A = QR$$

$$Q = \tilde{Q}^T = (P_{n-1} \cdots P_r \cdots P_2P_1)^T = P_1P_2 \cdots P_r \cdots P_{n-1}.$$

Algoritmul Householder

Pasul r:
La intrarea în pasul r matricea A are forma:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1r} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2r} & \cdots & a_{2n} \\ 0 & 0 & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{rr} & \cdots & a_{rn} \\ 0 & 0 & \cdots & a_{r+1r} & \cdots & a_{r+1n} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{ir} & \cdots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{nr} & \cdots & a_{nn} \end{pmatrix}$$

Algoritmul Householder

Aşadar, pasul r constă în:

$$A := P_r A$$

$$P_r = I_n - 2v^r (v^r)^T,$$

unde $P_r \in \mathbb{R}^{n\times n}$, iar vectorul $v_r \in \mathbb{R}^n$, $\|v^r\|_2 = 1$, se alege astfel ca matricea A să aibă şi coloana r în formă superior triunghiulară

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1r} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2r} & \cdots & a_{2n} \\ 0 & 0 & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{rr} & \cdots & a_{rn} \\ 0 & 0 & \cdots & 0 & \cdots & a_{r+1n} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & a_{nn} \end{pmatrix}.$$

Calculul matricei P_r

Pentru simplitate vom nota $P_r = P$, $v^r = v$.

$$A\mathbf{e}_r = \begin{pmatrix} a_{1r} \\ a_{2r} \\ \vdots \\ a_{r-1r} \\ a_{rr} \\ a_{r+1r} \\ \vdots \\ a_{ir} \\ \vdots \\ a_{nr} \end{pmatrix} \longrightarrow (PA)\mathbf{e}_r = \overline{A}\mathbf{e}_r = \begin{pmatrix} \bar{a}_{1r} = a_{1r} \\ \bar{a}_{2r} = a_{2r} \\ \vdots \\ \bar{a}_{r-1r} = a_{r-1r} \\ \bar{a}_{rr} = k \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Algoritmul Householder

Aplicând proprietatea matricelor ortogonale:

$$Q \in \mathbb{R}^{n\times n}, \qquad \mathbf{x} \in \mathbb{R}^n, \qquad \|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2$$

pentru matricea $Q = P$ și vectorul $\mathbf{x} = A\mathbf{e}_r$ avem:

$$\|PA\mathbf{e}_r\|_2^2 = a_{1r}^2 + a_{2r}^2 + \ldots + a_{r-1r}^2 + k^2$$

$$\|A\mathbf{e}_r\|_2^2 = a_{1r}^2 + a_{2r}^2 + \ldots + a_{r-1r}^2 + a_{rr}^2 + a_{r+1r}^2 + \ldots + a_{ir}^2 + \ldots + a_{nr}^2$$

Din relaţia de mai sus rezultă

$$k^2 = \sigma = a_{rr}^2 + a_{r+1r}^2 + \cdots + a_{ir}^2 + \cdots + a_{nr}^2 = \sum_{i=r}^{n} a_{ir}^2 \Rightarrow k = \pm\sqrt{\sigma}$$

Algoritmul Householder

Determinarea vectorului $\mathbf{v}$ ce defineşte matricea P:

$$\begin{aligned}(PA)\mathbf{e}_r &= (I_n - 2\mathbf{v}\mathbf{v}^T)(A\mathbf{e}_r) = A\mathbf{e}_r - 2(\mathbf{v}\mathbf{v}^T)(A\mathbf{e}_r) \\ &= A\mathbf{e}_r - 2\mathbf{v}(\mathbf{v}^T(A\mathbf{e}_r)) = A\mathbf{e}_r - (2\alpha)\mathbf{v} = A\mathbf{e}_r - \mathbf{u}\end{aligned}$$

unde cu α și $\mathbf{u}$ am notat:

Algoritmul lui Householder

$$\alpha := \mathbf{v}^T(A\mathbf{e}_r) = ((A\mathbf{e}_r), \mathbf{v})_{\mathbb{R}^n} = \left(\begin{pmatrix} a_{1r} \\ a_{2r} \\ \vdots \\ a_{ir} \\ \vdots \\ a_{nr} \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{pmatrix}\right)_{\mathbb{R}^n}$$

$$= a_{1r}v_1 + a_{2r}v_2 + \cdots + a_{ir}v_i + \cdots + a_{nr}v_n.$$

Algoritmul lui Householder

$$\mathbf{u} := (2\alpha)\mathbf{v} = A\mathbf{e}_r - (PA)\mathbf{e}_r = \begin{pmatrix} a_{1r} \\ a_{2r} \\ \vdots \\ a_{r-1r} \\ a_{rr} \\ a_{r+1r} \\ \vdots \\ a_{ir} \\ \vdots \\ a_{nr} \end{pmatrix} - \begin{pmatrix} a_{1r} \\ a_{2r} \\ \vdots \\ a_{r-1r} \\ k \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ a_{rr} - k \\ a_{r+1r} \\ \vdots \\ a_{ir} \\ \vdots \\ a_{nr} \end{pmatrix}$$

Algoritmul lui Householder

Cu aceste notații, matricea P devine:

$$P = I_n - 2\mathbf{v}\mathbf{v}^T = I_n - 2\left(\frac{1}{2\alpha}\right)\mathbf{u}\left(\frac{1}{2\alpha}\right)\mathbf{u}^T = I_n - \frac{1}{2\alpha^2}\mathbf{u}\mathbf{u}^T$$
$$= I_n - \frac{1}{\beta}\mathbf{u}\mathbf{u}^T,$$

unde $\beta = 2\alpha^2$.
Pentru a cunoaște matricea P trebuie să mai determinăm constanta β. Din condiția:

$$\|\mathbf{v}\|_2^2 = 1 \implies \left\|\frac{1}{2\alpha}\mathbf{u}\right\|_2^2 = 1 \implies \frac{1}{4\alpha^2}\|\mathbf{u}\|_2^2 = 1 \implies 2\beta = \|\mathbf{u}\|_2^2.$$

Algoritmul lui Householder

$$\|\mathbf{u}\|_2^2 = \left\| \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_{rr} - k \\ a_{r+1r} \\ \vdots \\ a_{ir} \\ \vdots \\ a_{nr} \end{pmatrix} \right\|_2^2 = (a_{rr} - k)^2 + a_{r+1r}^2 + \cdots + a_{ir}^2 + \cdots + a_{nr}^2$$

$$= a_{rr}^2 + a_{r+1r}^2 + \cdots + a_{ir}^2 + \cdots + a_{nr}^2 - 2ka_{rr} + k^2$$

$$= \sigma - 2ka_{rr} + \sigma = 2(\sigma - ka_{rr})$$

de unde obţinem

$$\beta = \sigma - ka_{rr}.$$

Algoritmul lui Householder

Vom alege semnul constantei k astfel încât β să fie cât mai mare posibil deoarece constanta β apare în operația de împărțire. Avem :

β " mare" $\rightarrow \beta = \sigma - k a_{rr} \geq \sigma \ (\sigma \geq 0) \rightarrow$ semn $k = -$semn a_{rr}

Ce înseamnă $\beta = 0$?

Algoritmul lui Householder

Ce înseamnă $\beta = 0$?

$$\beta = \frac{1}{2}\|\mathbf{u}\|_2^2 = 0 \longrightarrow \|\mathbf{u}\|_2^2 = 0 \longrightarrow \mathbf{u} = \mathbf{0}$$

$$\longrightarrow a_{rr} = k, a_{r+1r} = 0, \ldots, a_{ir} = 0, \ldots, a_{nr} = 0.$$

Cum $a_{rr} = k$ şi semn $k = -$semn a_{rr} obținem:

$$a_{ir} = 0, \quad \forall i = r, \ldots, n,$$

adică avem coloana r deja în formă superior triunghiulară, se poate trece la pasul următor. În acest caz matricea A este singulară.

Algoritmul lui Householder

Ne interesează cum se efectuează operația $A = P_r A$ fără a face înmulțire matricială.
Vom pune în evidentă schimbările în raport cu coloanele.

$$
\begin{aligned}
(PA)\mathbf{e}_j = &\ \text{noua col. } j \text{ a matr. } A = \left(I_n - \frac{1}{\beta}\mathbf{u}\mathbf{u}^T\right)(A\mathbf{e}_j) \\
&= A\mathbf{e}_j - \frac{1}{\beta}(\mathbf{u}\mathbf{u}^T)(A\mathbf{e}_j) = A\mathbf{e}_j - \frac{1}{\beta}\mathbf{u}(\mathbf{u}^T(A\mathbf{e}_j)) \\
&= A\mathbf{e}_j - \frac{\gamma_j}{\beta}\mathbf{u}.
\end{aligned}
$$

Algoritmul lui Householder

$$\gamma_j := \mathbf{u}^T(A\mathbf{e}_j) = \left(\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{rj} \\ a_{r+1j} \\ \vdots \\ a_{nj} \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ a_{rr}-k \\ a_{r+1r} \\ \vdots \\ a_{nr} \end{pmatrix}\right)_{\mathbf{R}^n}$$

$$= a_{rj}(a_{rr}-k) + \cdots + a_{ij}a_{ir} + \cdots + a_{nj}a_{nr}$$

$$= \sum_{i=1}^{n} u_i a_{ij} = \sum_{i=r}^{n} u_i a_{ij}$$

$$(u_i = 0,\ i = 1, \ldots, r-1,\ u_r = a_{rr}-k, u_i = a_{ir},\ i = r+1, \ldots, n)$$

Algoritmul lui Householder

Noua coloană j se obține din vechea coloană j din care scădem vectorul $\mathbf{u}$ înmulțit cu constanta γ_j.
Ne interesează ca primelor $(r-1)$ coloane să nu li se schimbe forma superior triunghiulară deja obținută.
Spre exemplu, pentru $j = 1, \ldots, r-1$, avem

$$\gamma_j := \mathbf{u}^T(A\mathbf{e}_j) = \left(\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{jj} \\ a_{j+1j} = 0 \\ \vdots \\ a_{rj} = 0 \\ \vdots \\ a_{nj} \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ a_{rr} - k \\ a_{r+1r} \\ \vdots \\ a_{nr} \end{pmatrix}\right)_{\mathbf{R}^n}$$

$$= a_{1j} \cdot 0 + \ldots + a_{jj} \cdot 0 + \cdots + 0 \cdot (a_{rr} - k) + \cdots + 0 \cdot a_{ir} + \cdots + 0 \cdot a_{nr} = 0$$

Algoritmul lui Householder

Trecerea de la matricea A la matricea P_rA:
Din faptul că $\gamma_j = 0, \quad j = 1, \ldots, r-1$, rezultă că primele $(r-1)$ coloane ale matricei A nu se schimbă când facem operația

$$A = P_rA,$$

rămân în formă superior triunghiulară.
Algoritmul de trecere de la matricea A la matricea P_rA este următorul:

$$(P_rA)\mathbf{e}_j = \begin{cases} A\mathbf{e}_j, & \text{pentru } j = 1, \ldots, r-1, \\ (a_{1r}, a_{2r}, \ldots, a_{r-1r}, k, 0, \ldots, 0)^T, & \text{pentru } j = r, \\ A\mathbf{e}_j - \dfrac{\gamma_j}{\beta}\mathbf{u}, & \text{pentru } j = r+1, \ldots, n. \end{cases}$$

Algoritmul lui Householder

$$\gamma_j = (A\mathbf{e}_j, \mathbf{u})_{\mathbf{R}^n} = \sum_{i=r}^{n} u_i a_{ij}, \quad \gamma_j = 0, \quad j = 1, \ldots, r-1$$

$$u_i = \begin{cases} 0, & i = 1, \ldots, r-1 \\ a_{rr} - k, & i = r \\ a_{ir}, & i = r+1, \ldots, n \end{cases}$$

Operația de transformare a vectorului termenilor liberi

$$\mathbf{b} := P_r \mathbf{b}$$

$$P_r\mathbf{b} = (I_n - \mathbf{u}\mathbf{u}^T)\mathbf{b} = \mathbf{b} - \frac{1}{\beta}(\mathbf{u}\mathbf{u}^T)\mathbf{b} = \mathbf{b} - \frac{1}{\beta}\mathbf{u}(\mathbf{u}^T\mathbf{b}) = \mathbf{b} - \frac{\gamma}{\beta}\mathbf{u}$$

$$\gamma = \mathbf{u}^T\mathbf{b} = (\mathbf{b}, \mathbf{u})_{\mathbb{R}^n} = \sum_{i=r}^{n} u_i b_i.$$

Algoritmul lui Householder

$$Q = I_n$$

for $r = 1, \ldots, n-1$

$$\begin{cases} \text{se calculează matricea } P_r \text{ (constanta } \beta \text{ și vectorul } \mathbf{u}) \\ A := P_r A \\ b := P_r \mathbf{b} \\ Q := P_r Q \end{cases}$$

La sfârșitul algoritmului, în matricea A vom avea matricea superior triunghiulară R, în vectorul $\mathbf{b}$ vom avea $Q^T \mathbf{b}^{init}$, $\mathbf{b}^{init}$ este vectorul inițial al termenilor liberi, iar matricea Q va conţine matricea Q^T din factorizarea QR a matricei A.

Algoritmul Householder (1/3)

$Q = I_n;$
for $r = 1, \ldots, n-1$ **do**
//**Construcția matricei** P_r (constanta β și vectorul u)

- $\sigma = \sum_{i=r}^{n} a_{ir}^2;$
- **if** $(\sigma \leq \varepsilon)$ **break;** $//r = r + 1 \Leftrightarrow P_r = I_n$ (A **singulară**)
- $k = \sqrt{\sigma};$
- **if** $(a_{rr} > 0)$ $k = -k;$
- $\beta = \sigma - k a_{rr};$
- $u_r = a_{rr} - k;\ u_i = a_{ir},\ i = r+1, \ldots n;$

Algoritmul Householder (2/3)

// $A = P_r * A$

//**transformarea coloanelor** $j = r+1, \ldots, n$

- **for** $j = r+1, \ldots, n$
  - $\gamma = (\gamma_i/\beta) = (A\mathbf{e}_j, \mathbf{u})/\beta = \left(\sum_{i=r}^{n} u_i a_{ij}\right)/\beta;$
  - **for** $i = r, \ldots, n$
    $a_{ij} = a_{ij} - \gamma * u_i;$

// **transformarea coloanei** r **a matricii** A

- $a_{rr} = k; a_{ir} = 0, \quad i = r+1, \ldots, n;$

// $b = P_r * b$

- $\gamma = (\gamma/\beta) = (\mathbf{b}, \mathbf{u})/\beta = \left(\sum_{i=r}^{n} u_i b_i\right)/\beta;$
- **for** $i = r, \ldots, n$
  $b_i = b_i - \gamma * u_i;$

Algoritmul Householder (3/3)

$// \; Q = P_r * Q$

- **for** $j = 1, \ldots, n$
  - $\gamma = (Q\mathbf{e}_j, \mathbf{u})/\beta = \left(\sum_{i=r}^{n} u_i q_{ij} \right) / \beta;$
  - **for** $i = r, \ldots, n$
    $q_{ij} = q_{ij} - \gamma * u_i;$

Numărul de operații efectuate

- **A** - (adunări, scăderi):

$$(n-1)+2n(n-1)+\frac{n(n-1)(2n-1)}{3} = \frac{(n-1)(n+1)(2n+3)}{3}$$

$$= \frac{2}{3}n^3 + O(n^2)$$

- **M** (înmulțiri, împărțiri):

$$4(n-1) + 3n(n-1) + \frac{n(n-1)(2n-1)}{6} = \frac{2}{3}n^3 + O(n^2)$$

- **R** (radicali): $n-1$

Descompunerea QR - Algoritmul lui Givens

Fie A o matrice reală pătratică de dimensiune n.
Presupunem că avem descompunerea:

$$A = QR$$

unde
Q este o matrice ortogonală, iar R este o matrice superior triunghiulară.
Atunci pentru sistemul liniar $A\mathbf{x} = \mathbf{b}$, avem:

$$A\mathbf{x} = \mathbf{b} \Leftrightarrow QR\mathbf{x} = \mathbf{b} \Leftrightarrow Q^T QR\mathbf{x} = Q^T\mathbf{b} \Leftrightarrow R\mathbf{x} = Q^T\mathbf{b}.$$

Descompunerea QR - Algoritmul lui Givens

În cazul algoritmului Givens, pentru a aduce sistemul $A\mathbf{x} = \mathbf{b}$ la forma $R\mathbf{x} = Q^T\mathbf{b}$ se folosesc *matricele de rotație*. Aceste matrice anulează elementele sub diagonala principală prin rotații în plan.
O matrice de rotaţie $R_{pq}(\theta) = (r_{ij})_{i,j=1,\dots,n}$ are următoarea formă:

$$R_{pq}(\theta) = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{array}{ccccccc} & & & p & q & & \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \dots & 0 & \dots & 0 \\ \vdots & & & & & & \\ 0 & 0 & \dots & c \dots & s & \dots & 0 \\ \vdots & & & 1 & & & \\ 0 & 0 & \dots & (-s) \dots & c & \dots & 0 \\ \vdots & & & & & & \\ 0 & 0 & \dots & 0 \dots & 0 & \dots & 1 \end{pmatrix} \end{array} \begin{array}{c} \\ \\ \\ \\ p \\ \\ q \\ \\ \\ \end{array}$$

Descompunerea QR - Algoritmul lui Givens

unde

$$r_{ij} = \begin{cases} 1 & \text{pentru } i = j, i \neq p, i \neq q; \\ c & \text{pentru } i = j, i = p, i = q; \\ s & \text{pentru } i = p, j = q; \\ -s & \text{pentru } i = q, j = p; \\ 0 & \text{în rest,} \end{cases}$$

iar $p, q \in \{1, \ldots n\}$, iar c şi s sunt două numere reale care satisfac relaţia $c^2 + s^2 = 1$.
Constantele c şi s pot fi alese astfel încât

$$c = \cos\theta, \quad s = \sin\theta.$$

Observaţie: Cum $c^2 + s^2 = 1$, se poate arăta uşor că matricea de rotaţie $R_{pq}(\theta)$ este ortogonală, adică

$$R_{pq}(\theta)R_{pq}^T(\theta) = R_{pq}^T(\theta)R_{pq}(\theta) = I_n.$$

Descompunerea QR - Algoritmul lui Givens

Calculul matricei B

$$B = R_{pq}(\theta)\, A$$

Matricea B se obține din A modificând doar liniile p și q. Fie

$$A_i = \mathbf{e}_i^T A \text{ - linia } i \text{ a matricei } A,$$

$$B_i = \mathbf{e}_i^T B \text{ - linia } i \text{ a matricei } B.$$

Calculul liniilor matricei B:

$$\begin{aligned} B_i &= A_i,\ i = 1, \ldots, n, i \neq p,\ i \neq q. \\ B_p &= cA_p + sA_q, \\ B_q &= -sA_p + cA_q. \end{aligned}$$

Descompunerea QR - Algoritmul lui Givens

$$b_{pj} = ca_{pj} + sa_{qj},\ j = 1, \ldots, n.$$

$$b_{qj} = -sa_{pj} + ca_{qj},\ j = 1, \ldots, n.$$

$$b_{ij} = a_{ij},\ \text{ în rest.}$$

Calculul matricei D, definită prin

$$D = AR_{pq}^T(\theta),$$

se obţine din A, modificând doar coloanele p şi q.

Notăm $A\mathbf{e}_j$, $D\mathbf{e}_j$ - coloana j a matricei A şi respectiv D.

Descompunerea QR - Algoritmul lui Givens

Coloanele matricei D:

$$
\begin{aligned}
&D_j = A_j,\ j = 1, \ldots, n,\ j \neq p,\ j \neq q.\\
&D\mathbf{e}_p = cA\mathbf{e}_p + sA\mathbf{e}_q,\\
&D\mathbf{e}_q = -sA\mathbf{e}_p + CA\mathbf{e}_q,\\
&d_{ip} = ca_{ip} + sa_{iq},\ i = 1, \ldots, n,\\
&d_{iq} = -sa_{ip} + ca_{iq},\ i = 1, \ldots, n,\\
&d_{ij} = a_{ij},\ \text{ în rest.}
\end{aligned}
$$

Algoritmul lui Givens se desfășoară în $(n-1)$ pași - la pasul r se transformă coloana r a matricei A în formă superior triunghiulară fără a modifica primele $(r-1)$ coloane.

Algoritmul Givens – Pasul 1

Pasul 1:

Intrare: matricea A, vectorul $\mathbf{b}$.
Ieșire: matricea $A^{(1)}$ (cu prima coloană în formă superior triunghiulară), vectorul $\mathbf{b}^{(1)}$.

Se efectuează următoarele operații de înmulțire cu matrice de rotație:

$$A^{(1)} = R_{1n}(\theta_{1n})R_{1n-1}(\theta_{1n-1})\cdots R_{13}(\theta_{13})R_{12}(\theta_{12})A$$

$$\mathbf{b}^{(1)} = R_{1n}(\theta_{1n})R_{1n-1}(\theta_{1n-1})\cdots R_{13}(\theta_{13})R_{12}(\theta_{12})\mathbf{b}.$$

Unghiurile θ_{1i} (respectiv constantele c_{1i} și s_{1i}) se aleg astfel încât elementul de pe poziția $(i, 1)$ din matricea rezultată să devină 0.

Algoritmul Givens – Pasul r

Pasul r:

Intrare: matricea $A^{(r-1)}$ (are primele $r-1$ coloane în formă superior triunghiulară), vectorul $\mathbf{b}^{(r-1)}$.

Ieșire: matricea $A^{(r)}$ (cu primele r coloane în formă superior triunghiulară), vectorul $\mathbf{b}^{(r)}$

La acest pas matricea $A^{(r-1)}$ și vectorul $\mathbf{b}^{(r-1)}$ se transformă astfel:

$$A^{(r)} = R_{rn}(\theta_{rn})R_{rn-1}(\theta_{r,n-1})\cdots R_{ri}(\theta_{ri})\cdots R_{rr+1}(\theta_{rr+1})A^{(r-1)},$$

$$\mathbf{b}^{(r)} = R_{rn}(\theta_{rn})R_{rn-1}(\theta_{rn-1})\cdots R_{rr+1}(\theta_{rr+1})\mathbf{b}^{(r-1)}.$$

unde elementele $c = c_{ri}$ şi $s = s_{ri}$ din matricele de rotație se aleg astfel ca după înmulțirea cu $R_{ri}(\theta_{ri})$, $i = r+1, \ldots, n$, elementrul (i, r) să devină 0.

Algoritmul Givens

Considerăm operația

$$B = R_{ri}(\theta_{ri})A,$$

unde θ_{ri} se alege astfel încât $b_{ir} = 0$.
Elementele matricei B se calculează astfel:

$$\begin{aligned}
&b_{rj} = c\,a_{rj} + s\,a_{ij},\\
&b_{ij} = -s\,a_{rj} + c\,a_{ij}, j = 1, \ldots, n\\
&(b_{kl} = a_{kl},\ \text{în rest})\\
&(j = r)\ \ b_{ir} = -sa_{rr} + ca_{ir}.
\end{aligned}$$

Alegerea coeficienților c și s

Cea mai simplă alegere pentru c și s, astfel încât să obținem $b_{ir} = 0$, este:

$$c = f \cdot a_{rr}, \qquad s = f \cdot a_{ir},$$

unde factorul f se alege astfel încât $c^2 + s^2 = 1$:

$$f = \frac{1}{\sqrt{a_{rr}^2 + a_{ir}^2}}.$$

Prin urmare,

$$c = \frac{a_{rr}}{\sqrt{a_{rr}^2 + a_{ir}^2}}, \qquad s = \frac{a_{ir}}{\sqrt{a_{rr}^2 + a_{ir}^2}}.$$

Dacă $\sqrt{a_{rr}^2 + a_{ir}^2} = 0 \Rightarrow a_{rr} = a_{ir} = 0 \Rightarrow c = 1, s = 0.$ $(R_{ir}(\theta) = I_n)$

Efectul rotației Givens asupra matricei

Elementul de pe poziția (i, r) este deja nul, deci nu avem ce schimba în matricea A.

Operația $B = R_{ri}(\theta_{ri})A$, nu afectează forma superior triunghiulară a primelor $(r-1)$ coloane.
În matricea B, aceste coloane rămân în continuare în formă superior triunghiulară.
Pentru $j = 1, \ldots, r-1$, cum $a_{rj} = 0, a_{ij} = 0$, avem:

$$b_{rj} = c\, a_{rj} + s\, a_{ij} = 0, \qquad b_{ij} = -s\, a_{rj} + c\, a_{ij} = 0.$$

Înmulțirea $B = R_{ri}(\theta_{ri})A$ modifică doar liniile r și i ale matricei B.

Rotația Givens nu modifică elementele nule deja obținute, ci doar anulează elementul de pe poziția (i, r).

Algoritmul lui Givens

Algoritmul lui Givens poate fi descris prin produsul matricilor de rotație astfel:

$$R_{nn-1}(\theta_{nn-1}) \cdots R_{rn}(\theta_{rn}) \cdots R_{rr+1}(\theta_{rr+1}) \cdots R_{1n}(\theta_{1n}) \cdots R_{12}(\theta_{12})A = R$$

Aplicând aceleași rotații asupra vectorului **b** obținem:

$$R_{nn-1}(\theta_{nn-1}) \cdots R_{rn}(\theta_{rn}) \cdots R_{rr+1}(\theta_{rr+1}) \cdots R_{1n}(\theta_{1n}) \cdots R_{12}(\theta_{12})\mathbf{b} = \overline{\mathbf{b}} = Q^T\mathbf{b}.$$

Notăm cu Q următoarea matrice:

$$Q = R_{n-1n}(\theta_{n-1n}) \cdots R_{rn}(\theta_{rn}) \cdots R_{rr+1}(\theta_{rr+1}) \cdots R_{1n}(\theta_{1n}) \cdots R_{12}(\theta_{12}).$$

Descompunerea QR prin rotații Givens

Matricea Q este ortogonală deoarece este produsul unor matrice ortogonale. ($Q^{-1} = Q^T$)
Descompunerea QR a matricei A este următoare:

$$QA = R \ \ (Q^{-1} = Q^T) \Rightarrow A = Q^T R = QR.$$

$$\begin{aligned} Q = Q^T &= (R_{n-1n}(\theta_{n-1n}) \cdots R_{rn}(\theta_{rn}) \cdots R_{1n}(\theta_{1n}) \cdots R_{12}(\theta_{12}))^T \\ &= R_{12}^T(\theta_{12}) \cdots R_{1n}^T(\theta_{1n}) \cdots R_{rn}^T(\theta_{rn}) \cdots R_{n-1n}^T(\theta_{n-1n}). \end{aligned}$$

Algoritmul lui Givens (pe scurt)

Pe scurt, *algoritmul lui Givens* este următorul:

$$
\begin{aligned}
&Q = I_n;\\
&\textbf{for}\ \ r = 1, \ldots, n-1\\
&\qquad \textbf{for}\ \ i = r+1, \ldots, n\\
&\qquad\qquad \begin{cases} A = R_{ri}(\theta_{ri}) * A;\\ \mathbf{b} = R_{ri}(\theta_{ri}) * \mathbf{b};\\ Q = R_{ri}(\theta_{ri}) * Q. \end{cases}
\end{aligned}
$$

Algoritmul lui Givens (1/2)

$Q = I_n;$

for r=1,. . . ,n-1

for i=r+1,. . . ,n

//construcţia matricii $R_{ri}(\theta_{ri})$ *- constantele* c şi s

- $f = \sqrt{a_{rr}^2 + a_{ir}^2};$
- **if** $(f \leq \varepsilon)$ $\{c = 1; s = 0;\}//R_{ri}(\theta_{ri}) = I$
  else $\{c = a_{rr}/f;\ s = a_{ir}/f;\}$
  $//A = R_{ri}(\theta_{ri}) * A$
- **for** $j = r + 1, \ldots, n$
  $$\begin{cases} a_{rj} = c * a_{rj} + s * a_{ij}; \\ a_{ij} = -s * a_{rj}^{vechi} + c * a_{ij}; \end{cases}$$
- $a_{ir} = 0;\ a_{rr} = f;$

Algoritmul lui Givens (2/2)

$//b = R_{ri}(\theta_{ri}) * b$

- $b_r = c * b_r + s * b_i;$
- $b_i = -s * b_r^{vechi} + c * b_i;$

$//Q = R_{ri}(\theta_{ri}) * Q$

- **for** $j = 1, \ldots, n$

$$\begin{cases} \tilde{q}_{rj} = c * \tilde{q}_{rj} + s * \tilde{q}_{ij}; \\ \tilde{q}_{ij} = -s * \tilde{q}_{rj}^{vechi} + c * \tilde{q}_{ij}; \end{cases}$$

Algoritmul lui Givens

La sfârșitul acestui algoritm,

- în matricea A se obține matricea superior triunghiulară R,
- în vectorul $\mathbf{b}$ se obține $Q\mathbf{b}^{\text{init}}$, unde $\mathbf{b}^{\text{init}}$ este vectorul termenilor liberi inițial,
- matricea Q va conține matricea Q^T din factorizarea QR a matricei A.

Numărul operaţiilor efectuate (exceptând calculul matricei Q) obţinem:

- **R** -radicali: $\dfrac{n(n-1)}{2}$
- **A** -adunări/scăderi: $\dfrac{n(n-1)(4n+7)}{6} = \dfrac{2}{3}n^3 + O(n^2)$;
- **M**-înmulţiri/împărţiri: $\dfrac{2n(n-1)(2n+5)}{3} = \dfrac{4}{3}n^3 + O(n^2)$.

Descompunerea QR – algoritmul Gram–Schmidt

Fie $A \in \mathbb{R}^{n\times n}$, cu $\det A \neq 0$.
Atunci matricea A admite factorizarea:

$$A = QR$$

unde

$$\begin{pmatrix} a^1 & a^2 & \cdots & a^n \end{pmatrix} = \begin{pmatrix} q^1 & q^2 & \cdots & q^n \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{pmatrix}. \quad (2)$$

unde

- $a^j = A\mathbf{e}_j$ - coloana j a matricei A;
- $q^j = Q\mathbf{e}_j$ - coloana j a matricei Q.

Descompunerea QR – algoritmul Gram–Schmidt

Relaţia (2) poate fi rescrisă astfel:

$$\begin{cases} r_{11}q^1 = a^1 \\ r_{12}q^1 + r_{22}q^2 = a^2 \\ \vdots \\ r_{1p}q^1 + \ldots + r_{jp}q^j + \ldots + r_{pp}q^p = a^p \\ \vdots \\ r_{1n}q^1 + \ldots + r_{jn}q^j + \ldots + r_{nn}q^n = a^n \end{cases} \tag{3}$$

Descompunerea QR – algoritmul Gram–Schmidt

Algoritmul de calcul al descompunerii QR cu metoda Gram-Schmidt se desfășoară în n pași, la fiecare pas calculându-se:

- coloana p din matricea R;
- coloana p din matricea Q.

Aşadar, avem

$$\det A \neq 0, \ A = QR, \ \text{unde } Q \text{ - ortogonală}$$

$$\Rightarrow \det R \neq 0 \ (r_{ii} \neq 0, \ \forall i)$$

Descompunerea QR – algoritmul Gram–Schmidt

Pasul 1:

Se folosește prima ecuație a sistemului (3)

$$r_{11}q^1 = a^1.$$

Se face produsul scalar al acestei ecuații cu vectorii q^1 şi a^1.
Se folosește proprietatea coloanelor matricelor ortogonale:

$$(q^i, q^j)_{\mathbb{R}^n} = \begin{cases} \|q^i\|_2^2 = 1, & \textit{pentru} i = j \\ 0, & \text{pentru } i \neq j \end{cases}$$

$$q^1 = \frac{1}{r_{11}} a^1 \quad \text{şi} \quad \|q^1\|_2 = 1.$$

$$r_{11} = \pm\|a^1\|_2, \qquad q^1 = \frac{1}{r_{11}} a^1 \ (r_{11} \neq 0 \text{ deoarece } \det A \neq 0).$$

Descompunerea QR – algoritmul Gram–Schmidt

Pasul r:

Se folosește ecuația p sistemului (3):

$$r_{1p}q^1 + \ldots + r_{jp}q^j + \ldots + r_{pp}q^p = a^p.$$

La acest pas se cunosc deja coloanele $q^1, q^2, \ldots, q^{p-1}$.

Se face, pe rând, produsul scalar al ecuației cu vectorii

$$q_1, q_2, \ldots, q_{p-1},\ q_p \text{ și } a_p.$$

$$\left(\sum_{k=1}^{p} r_{kp}q^k, q^j\right)_{\mathbb{R}^n} = \left(a^p, q^j\right)_{\mathbb{R}^n},\ j = 1, \ldots, p-1.$$

Descompunerea QR – algoritmul Gram–Schmidt

Pasul r:

$$\left(\sum_{k=1}^{p} r_{kp} q^k, q^j\right)_{\mathbb{R}^n} = \sum_{k=1, k\neq j}^{p} r_{kp} \left(q^k, q^j\right)_{\mathbb{R}^n} + r_{jp} \left(q^j, q^j\right)_{\mathbb{R}^n} = r_{jp}$$

$$r_{jp} = \left(a^p, q^j\right)_{\mathbb{R}^n}, \; j = 1, \ldots, p-1.$$

Aşadar, avem:

$$q^p = \frac{1}{r_{pp}} \left(a^p - r_{1p} q^1 - \ldots - r_{p-1p} q^{p-1}\right)$$

$$\|q^p\|_2^2 = 1 = \frac{1}{r_{pp}^2} \|(a^p - r_{1p} q^1 - \ldots - r_{p-1p} q^{p-1})\|_2^2$$

Descompunerea QR – algoritmul Gram–Schmidt

Pasul r:

Aşadar, obţinem

$$r_{pp} = \pm \|a^p - r_{1p}q^1 - \ldots - r_{p-1p}q^{p-1}\|_2,$$

$$q^p = \frac{1}{r_{pp}}\left(a^p - r_{1p}q^1 - \ldots - r_{p-1p}q^{p-1}\right) = \frac{1}{r_{pp}}\left(a^p - \sum_{j=1}^{p-1} r_{jp}q^j\right).$$

Algoritmul Gram–Schmidt modificat

$$
\begin{aligned}
&\textbf{for } i = 1, \ldots, n \\
&\qquad v^i = a^i \\
&\textbf{for } i = 1, \ldots, n \\
&\qquad r_{ii} = \|v^i\|_2; \\
&\qquad q^i = (1/r_{ii})v^i; \\
&\qquad \textbf{for } j = i+1, \ldots n \\
&\qquad\qquad r_{ij} = (q^i, v^j); \\
&\qquad\qquad v^j = v^j - r_{ij}q^i;
\end{aligned}
$$

Numărul de operații:

$$\mathbf{M}: \frac{(n^3 + 3n^2)}{2} \qquad \mathbf{A}: \frac{(n^3 + n^2 - 2)}{2}.$$