

Algoritmul QR cu deplasare explicită

Algoritmul **QR**, considerat drept unul dintre rezultatele cele mai remarcabile ale calculului numeric matriceal (elaborat, în forma sa cea mai evoluată, independent, de către V.N. Kublanovskaia [20] și J.G.F. Francis [9]), este, în esență, o procedură de construcție iterativă a unui șir de matrice ortogonal asemenea cu matricea inițială și rapid convergent către forma Schur reală.

Așa cum am mai precizat, ne vom ocupa practic în exclusivitate de calculul valorilor proprii pentru matrice reale din trei motive:

- matricele reale sunt cel mai des întâlnite în aplicații;
- cazul matricelor complexe poate fi redus la cel al matricelor reale de ordin dublu (vezi problema rezolvată R9.13.);
- versiunea complexă necesită introducerea mijloacelor adecvate (transformări unitare, reflectori și rotații complexe etc.) pentru care nu dispunem de spațiul tipografic necesar; precizăm totuși că ideile sunt absolut similare iar cititorul interesat poate consulta [5], [13], [29].

Toate versiunile algoritmului **QR** sunt organizate în două etape:

1. *etapa directă*, de reducere a matricei date la forma superior Hessenberg prin transformări ortogonale de asemănare (algoritmul **HQ** din secțiunea precedentă);
2. *etapa iterativă*, de construcție recurentă a unui șir de matrice convergent către forma Schur reală.

În cele ce urmează vom considera parcursă prima etapă și numai la asamblarea întregului algoritm vom evidenția în mod explicit reducerea la forma superior Hessenberg.

Partea iterativă a algoritmului **QR** prezintă patru versiuni și anume

- a) cu deplasare explicită cu pași simpli,
- b) cu deplasare explicită cu pași dubli,
- c) cu deplasare implicită cu pași simpli,
- d) cu deplasare implicită cu pași dubli,

dintre care numai ultima este utilizată ca bază a programelor profesionale de calcul al valorilor și vectorilor proprii. Motivele constau în faptul că versiunile cu pas simplu nu asigură convergența în prezența valorilor proprii complexe iar versiunea cu pași dubli cu deplasare explicită este inefficientă, ascunzând, într-un mod cu totul insidios (!), calcule redundante. Totuși, pentru claritatea expunerii (de exemplu, pasul dublu este, în esență, concatenarea a doi pași simpli succesivi), vom parcurge, la nivelul ideilor, relațiilor și al eventualelor scheme de calcul, toate versiunile; scrierea detaliată a algoritmului formal o vom face numai pentru versiunea cea mai evoluată și anume cea cu deplasare implicită cu pași dubli.

Algoritmul QR cu deplasare explicită cu pași simpli

Fie matricea superior Hessenberg ireductibilă $\mathbf{H} \in \mathbb{R}^{n \times n}$ (dacă matricea $\mathbf{H}$ este reductibilă, atunci problema fie își reduce dimensiunea, fie se sparge în două subprobleme de dimensiuni mai mici). Algoritmul **QR** cu deplasare explicită cu pas simplu este definit de următoarea schemă de calcul

1. $\mathbf{H}_0 = \mathbf{H}$
2. pentru $k = 0, 1, 2, \dots$
 1. $\mathbf{H}_k - \mu_k \mathbf{I}_n = \mathbf{Q}_k \mathbf{R}_k$
 2. $\mathbf{H}_{k+1} = \mathbf{R}_k \mathbf{Q}_k + \mu_k \mathbf{I}_n$

care construiește iterativ șirul matriceal

$$\mathbf{H}_0, \mathbf{H}_1, \dots, \mathbf{H}_k, \mathbf{H}_{k+1}, \dots,$$

numit șirul **QR**. Instrucțiunile din ciclu definesc o iterație sau *un pas QR*. În această secțiune vom descrie succint proprietățile șirului **QR** și vom da o justificare calităților sale excepționale de convergență.

După cum se observă, un pas **QR** este format dintr-o factorizare **QR** (vezi capitolul 2) a matricei curente H_k având elementele diagonale "deplasate" cu scalarul μ_k și apoi matricea succesor H_{k+1} se obține din multiplicarea, în ordine inversă, a rezultatelor factorizării, urmată de refacerea deplasării. Principalele proprietăți ale acestor transformări sunt sintetizate în următoarea teoremă.

Teorema 1 a) Matricele H_k și H_{k+1} și, prin urmare, toate matricele șirului **QR**, sunt ortogonal asemenea.
b) Un pas **QR** conservă structura superior Hessenberg. Prin urmare dacă matricea inițială este superior Hessenberg, atunci toate matricele șirului **QR** sunt superior Hessenberg.

Demonstrație. a) Din prima relație a pasului **QR** avem $\mathbf{R}_k = \mathbf{Q}_k^T(\mathbf{H}_k - \mu_k \mathbf{I}_n)$, de unde

$$\mathbf{H}_{k+1} = \mathbf{R}_k \mathbf{Q}_k + \mu_k \mathbf{I}_n = \mathbf{Q}_k^T(\mathbf{H}_k - \mu_k \mathbf{I}_n) \mathbf{Q}_k + \mu_k \mathbf{I}_n = \mathbf{Q}_k^T \mathbf{H}_k \mathbf{Q}_k.$$

În consecință, toate matricele H_k din șirul **QR** sunt ortogonal asemenea cu matricea inițială H .

$$\mathbf{H}_k = \tilde{\mathbf{Q}}_k^T \mathbf{H} \tilde{\mathbf{Q}}_k, \quad k = 0, 1, 2, \dots$$

unde matricea cumulată a transformărilor ortogonale

$$\tilde{\mathbf{Q}}_k = \mathbf{Q}_0 \mathbf{Q}_1 \dots \mathbf{Q}_{k-1}$$

se poate obține iterativ cu relațiile

$$\tilde{\mathbf{Q}}_0 = \mathbf{Q}_0 = \mathbf{I}_n, \quad \tilde{\mathbf{Q}}_{k+1} = \tilde{\mathbf{Q}}_k \mathbf{Q}_k, \quad k = 0, 1, 2, \dots$$

b) Dacă H_k este o matrice superior Hessenberg, atunci în factorizarea **QR** din instrucțiunea 2.1 matricea ortogonală Q_k rezultă superior Hessenberg (demonstrați!). Cum produsul unei matrice superior triunghiulare cu o matrice superior Hessenberg este superior Hessenberg (demonstrați!), rezultă că H_{k+1} este o matrice superior Hessenberg. Teorema este demonstrată.

Faptul că toate matricele din șirul **QR** sunt superior Hessenberg are urmări benefice pentru eficiența algoritmului **QR** (complexitatea unei iterații se reduce de la $O(n^3)$ la $O(n^2)$).

În continuare expunem o proprietate extrem de importantă a șirului **QR**, proprietate care, nefiind adevărată întotdeauna, nu este un fapt matematic și, prin urmare, nu a putut să fie inclusă în teorema de mai sus. Proprietatea s-a dovedit adevărată practic în toate aplicațiile concrete dar a fost invalidată pe exemple special construite în acest scop. Remarcăm această situație paradoxală în care un rezultat fără valoare universală stă la baza celei mai performante metode de calcul al valorilor proprii ale unei matrice. Enunțul proprietății este următorul.

c) Printr-o alegere adecvată a deplasării μ_k șirul **QR** converge, aproape întotdeauna, către forma Schur a matricei H .

Justificare. Vom arăta mai mult, și anume că, printr-o alegere corespunzătoare a deplasărilor μ_k , această convergență poate fi făcută extrem de rapidă. În fapt, trebuie arătat că unele din elementele subdiagonale ale matricei H_k se anulează asimptotic când $k \rightarrow \infty$. Pentru simplificare, considerăm că matricea H are numai valori proprii reale, respectiv trebuie anulate asimptotic toate elementele subdiagonale (în caz contrar trebuie să luăm în considerație deplasări complexe).

$$\mathbf{R}_k = \mathbf{Q}_k^T (\mathbf{H}_k - \mu_k \mathbf{I}_n) = \mathbf{Q}_k^T (\tilde{\mathbf{Q}}_k^T \mathbf{H} \tilde{\mathbf{Q}}_k - \mu_k \mathbf{I}_n) = \tilde{\mathbf{Q}}_{k+1}^T (\mathbf{H} - \mu_k \mathbf{I}_n) \tilde{\mathbf{Q}}_k,$$

de unde avem

$$\tilde{\mathbf{Q}}_{k+1} \mathbf{R}_k = (\mathbf{H} - \mu_k \mathbf{I}_n) \cdot \tilde{\mathbf{Q}}_k.$$

Ținând seama de faptul că matricea R_k este superior triunghiulară, prima coloană a egalității devine (în cazul $\mu_k \notin \lambda(\mathbf{H})$)

$$\tilde{\mathbf{q}}_1^{(k+1)} = \frac{1}{r_{11}^{(k)}} (\mathbf{H} - \mu_k \mathbf{I}_n) \cdot \tilde{\mathbf{q}}_1^{(k)},$$

unde $\tilde{\mathbf{q}}_1^{(k)} = \tilde{\mathbf{Q}}_k \mathbf{e}_1$. Relația arată că prima coloană a matricei ortogonale $\tilde{\mathbf{Q}}_k$ se supune iterației metodei puterii și, în consecință, tinde către vectorul propriu unitar asociat valorii proprii dominante a matricei $\mathbf{A} - \mu_\infty \mathbf{I}_n$ (dacă inițializarea $\tilde{\mathbf{q}}_1^{(0)}$ are componentă nenulă pe această direcție, ceea ce, generic, este satisfăcut), respectiv

$$\tilde{\mathbf{Q}}_\infty = \lim_{k \rightarrow \infty} \tilde{\mathbf{Q}}_k = \begin{bmatrix} \mathbf{x}_1 & \mathbf{Y} \end{bmatrix}.$$

Conform procedurii de deflație rezultă că

$$\lim_{k \rightarrow \infty} h_{21}^{(k)} = 0,$$

adică matricea $\mathbf{H}_k$ tinde către forma Schur în prima coloană punând în evidență pe poziția (1,1), o valoare proprie (reală) a matricei $\mathbf{H}$.

Transpunând relația avem $\mathbf{R}_k^T \tilde{\mathbf{Q}}_{k+1}^T = \tilde{\mathbf{Q}}_k^T (\mathbf{H}^T - \mu_k \mathbf{I}_n)$, sau $\tilde{\mathbf{Q}}_k \mathbf{R}_k^T = (\mathbf{H}^T - \mu_k \mathbf{I}_n) \tilde{\mathbf{Q}}_{k+1}$. Luând egalitatea ultimelor coloane din ultima relație obținem

$$(\mathbf{H}^T - \mu_k \mathbf{I}_n) \cdot \tilde{\mathbf{q}}_n^{(k+1)} = r_{nn} \tilde{\mathbf{q}}_n^{(k)},$$

care arată că ultima coloană a matricei $\tilde{\mathbf{Q}}_k$ se supune iterației puterii inverse pentru matricea $\mathbf{H}^T$ cu deplasarea curentă de accelerare a convergenței μ_k . În consecință, coloana $\tilde{\mathbf{q}}_n^{(k)}$ tinde către un vector propriu al matricei $\mathbf{H}^T$ și anume către cel asociat valorii proprii $\lambda = \mu_\infty = \lim_{k \rightarrow \infty} \mu_k$, cea mai apropiată de aproximația inițială. Această convergență este cu atât mai rapidă cu cât deplasarea curentă este mai apropiată de o valoare proprie a matricei H . În consecință, analog cu (110) și simultan cu ea, avem

$$\tilde{\mathbf{Q}}_\infty = \lim_{k \rightarrow \infty} \tilde{\mathbf{Q}}_k = \begin{bmatrix} \mathbf{z} & \mathbf{u}_n \end{bmatrix},$$

unde $\mathbf{u}_n$ este un vector propriu unitar al matricei $\mathbf{H}^T$. Rezultă că matricea limită a șirului $\mathbf{Q}\mathbf{R}$ este:

$$\mathbf{H}_\infty = \tilde{\mathbf{Q}}_\infty^T \mathbf{H} \tilde{\mathbf{Q}}_\infty = (\tilde{\mathbf{Q}}_\infty^T \mathbf{H}^T \tilde{\mathbf{Q}}_\infty)^T.$$

Dar, întrucât $\lambda(\mathbf{H}^T) = \lambda(\mathbf{H})$ și $\mathbf{H}^T \mathbf{u}_n = \lambda_n \mathbf{u}_n$, rezultă relația

$$\tilde{\mathbf{Q}}_{\infty}^T \mathbf{H}^T \tilde{\mathbf{Q}}_{\infty} = \begin{bmatrix} \mathbf{Z}^T \\ \mathbf{u}_n^T \end{bmatrix} \cdot \mathbf{A}^T \cdot \begin{bmatrix} \mathbf{Z} & \mathbf{u}_n \end{bmatrix} = \begin{bmatrix} \mathbf{Z}^T \mathbf{H}^T \mathbf{Z} & 0 \\ \mathbf{u}_n^T \mathbf{H}^T \mathbf{Z} & \lambda_n \end{bmatrix},$$

care pune în evidență o deflație în ultima coloană a matricei $\mathbf{H}^T$ sau, echivalent, o deflație în ultima linie a matricei $\mathbf{H}$. Ținând seama de faptul că procesul iterativ are loc cu conservarea structurii superior Hessenberg rezultă că în matricea $\mathbf{H}_k$ din șirul $\mathbf{QR}$ avem

$$\lim_{k \rightarrow \infty} h_{n, n-1}^{(k)} = 0$$

cu o viteză mare, caracteristică metodei puterii inverse. Din (115) reiese că

$$\lim_{k \rightarrow \infty} h_{n, n}^{(k)} = \lambda_n$$

ceea ce înseamnă că $h_{nn}^{(k)}$ reprezintă o aproximare din ce în ce mai bună a valorii proprii λ_n . În consecință, alegerea deplasării

$$\mu_k = h_{n, n}^{(k)}$$

asigură o convergență pătratică (adică excelentă) pentru anularea elementului $h_{n, n-1}$.

Am arătat deci că șirul $\mathbf{QR}$ asigură o deflație iterativă în prima coloană și, cu alegerea (117) a deplasării, una foarte rapidă în ultima linie. În practică se constată chiar mai mult, și anume, reducerea simultană, cu viteze diferite, a tuturor modulelor elementelor subdiagonale, fapt care explică remarcabilele proprietăți de convergență ale algoritmului. Cu toate acestea există situații “patologice”, dificil de caracterizat matematic, în care șirul $\mathbf{QR}$ nu converge (vezi [5]). Cu această observație încheiem justificarea proprietății c).

Trunchierea procesului iterativ al șirului $\mathbf{QR}$ se face progresiv, anulându-se efectiv elementele subdiagonale convergente către zero în momentul satisfacerii unui criteriu de forma

$$|h_{p, p-1}^{(k)}| \leq \varepsilon \cdot \|\mathbf{H}_k\|$$

sau, mai simplu:

$$|h_{p, p-1}^{(k)}| \leq \varepsilon \cdot \left(|h_{p-1, p-1}^{(k)}| + |h_{pp}^{(k)}| \right),$$

unde ε este o toleranță precizată, iar $\|\cdot\|$ o normă matriceală oarecare. Anularea unuia sau mai multor elemente subdiagonale reduce, evident, dimensiunea problemei de deflație iterativă, spargând eventual problema curentă în două sau mai multe probleme de aceeași natură.

Întrucât, așa cum s-a mai menționat, implementările profesionale ale algoritmului $\mathbf{QR}$ merg pe o cale specifică (prezentată mai departe), recomandăm cititorului, ca un exercițiu util, să elaboreze algoritmul de implementare a unui pas $\mathbf{QR}$ definit în (101) cu alegerea (117) a deplasării. Se recomandă efectuarea calculelor pe loc, în tabloul matricei inițiale $\mathbf{H} \leftarrow \mathbf{H}_k$. Pentru factorizarea $\mathbf{QR}$ a matricei $\mathbf{H} - \mu \mathbf{I}_n$ din (101) se recomandă utilizarea unei secvențe de rotații plane. Calculul efectiv al matricei de transformare curente $\mathbf{Q} = \mathbf{Q}_k$ nu este necesar putându-se utiliza forma sa factorizată

$$\mathbf{Q} = \mathbf{P}_{12} \mathbf{P}_{23} \cdots \mathbf{P}_{n-1, n}.$$

Astfel schema de calcul propusă pentru suprascrierea matricei $\mathbf{H} = \mathbf{H}_k$ cu matricea succesor $\mathbf{H}_{k+1}$ din șirul $\mathbf{QR}$ are următoarea formă.

1. $\mu = h_{nn}$
2. $H \leftarrow H - \mu I_n$
3. Pentru $j = 1 : n - 1$
 1. Se determină rotația plană $P_{j,j+1}$ astfel încât
$$(P_{j,j+1}^T H)_{j+1,j} = 0.$$
 2. $H \leftarrow P_{j,j+1}^T H$
4. Pentru $j = 1 : n - 1$
 1. $H \leftarrow H P_{j,j+1}$
5. $H \leftarrow H + \mu I_n$

Algoritmul **QR** cu deplasare explicită funcționează foarte bine în cazul matricelor cu spectru real, existența valorilor proprii complexe crează probleme serioase în evidențierea asimptotică a blocurilor 2×2 din forma Schur reală.

9.5.2. Algoritmul QR cu deplasare explicită cu pași dubli

Pentru depășirea dificultăților legate de absența convergenței șirului **QR** creat de utilizarea pașilor simpli atunci când matricea are valori proprii complexe se adoptă așa numita strategie a *pașilor dubli QR* care comprimă într-o singură iterație doi pași simpli **QR** succesivi. Pentru a deduce procesarea aferentă unui pas dublu curent observăm că din ecuațiile a doi pași simpli **QR** consecutivi descriși, conform relațiilor (101), de

$$\begin{aligned} \mathbf{H}_k - \mu_k \mathbf{I}_n &= \mathbf{Q}_k \mathbf{R}_k \\ \mathbf{H}_{k+1} &= \mathbf{R}_k \mathbf{Q}_k + \mu_k \mathbf{I}_n \\ \mathbf{H}_{k+1} - \mu_{k+1} \mathbf{I}_n &= \mathbf{Q}_{k+1} \mathbf{R}_{k+1} \\ \mathbf{H}_{k+2} &= \mathbf{R}_{k+1} \mathbf{Q}_{k+1} + \mu_{k+1} \mathbf{I}_n \end{aligned}$$

rezultă mai întâi $\mathbf{H}_{k+1} = \mathbf{Q}_k^T \mathbf{H}_k \mathbf{Q}_k$, relație care, introdusă în ecuația a treia din (122), conduce la $\mathbf{Q}_k^T \mathbf{H}_k \mathbf{Q}_k - \mu_{k+1} \mathbf{I}_n = \mathbf{Q}_{k+1} \mathbf{R}_{k+1}$. Înmulțind această relație la stânga cu $\mathbf{Q}_k$ și la dreapta cu $\mathbf{R}_k$ obținem $\mathbf{H}_k \mathbf{Q}_k \mathbf{R}_k - \mu_{k+1} \mathbf{Q}_k \mathbf{R}_k = \mathbf{Q}_k \mathbf{Q}_{k+1} \mathbf{R}_{k+1} \mathbf{R}_k$, ecuație care permite obținerea lui $\mathbf{H}_{k+2}$ direct din $\mathbf{H}_k$ pe baza relațiilor

$$\begin{aligned} (\mathbf{H}_k - \mu_{k+1} \mathbf{I}_n) (\mathbf{H}_k - \mu_k \mathbf{I}_n) &= \mathbf{Q}_k \mathbf{Q}_{k+1} \mathbf{R}_{k+1} \mathbf{R}_k \\ \mathbf{H}_{k+2} &= (\mathbf{Q}_k \mathbf{Q}_{k+1})^T \mathbf{H}_k \mathbf{Q}_k \mathbf{Q}_{k+1} \end{aligned}$$

care, la rândul lor, se mai scriu sub forma

$$\begin{aligned} \mathbf{M} &= \mathbf{H}_k^2 - \mathbf{s}_k \mathbf{H}_k + \mathbf{p}_k \mathbf{I}_n = \bar{\mathbf{Q}}_k \bar{\mathbf{R}}_k \\ \mathbf{H}_{k+2} &= \bar{\mathbf{Q}}_k^T \mathbf{H}_k \bar{\mathbf{Q}}_k, \end{aligned}$$

unde $\mathbf{s}_k = \mu_k + \mu_{k+1}$ și $\mathbf{p}_k = \mu_k \mu_{k+1}$. Evident, matricea $\bar{\mathbf{Q}}_k = \mathbf{Q}_k \mathbf{Q}_{k+1}$ este ortogonală iar matricea $\bar{\mathbf{R}}_k = \mathbf{R}_{k+1} \mathbf{R}_k$ este superior triunghiulară.

Ceea ce este important constă în faptul că într-un pas dublu **QR** apare ca deosebit de naturală alegerea celor două deplasări μ_k , μ_{k+1} egale cu cele două valori proprii (posibil complex conjugate) ale blocului 2×2 din dreapta jos a matricei curente $\mathbf{H}_k$, i.e. ale matricei

$$\mathbf{G}_k = \begin{bmatrix} \mathbf{h}_{n-1, n-1}^{(k)} & \mathbf{h}_{n-1, n}^{(k)} \\ \mathbf{h}_{n, n-1}^{(k)} & \mathbf{h}_{n, n}^{(k)} \end{bmatrix}.$$

Aceste deplasări apar în expresia matricei $\mathbf{M}$ numai în forma sumei și produsului

$$\begin{aligned} \mathbf{s}_k &= \mathbf{u}_k + \mathbf{u}_{k+1} = \mathbf{h}_{n-1, n-1}^{(k)} + \mathbf{h}_{n, n}^{(k)} \\ \mathbf{p}_k &= \mathbf{u}_k \mathbf{u}_{k+1} = \mathbf{h}_{n-1, n-1}^{(k)} \mathbf{h}_{n, n}^{(k)} - \mathbf{h}_{n, n-1}^{(k)} \mathbf{h}_{n-1, n}^{(k)}, \end{aligned}$$

care sunt *reale*, inclusiv în cazul în care blocul $\mathbf{G}_k$ are valori proprii *complexe*. În consecință, matricea $\mathbf{M}$ este reală, produsul $\bar{\mathbf{Q}}_k \bar{\mathbf{R}}_k$ reprezintă factorizarea **QR** a matricei $\mathbf{M}$ cu $\bar{\mathbf{Q}}_k$ și $\bar{\mathbf{R}}_k$ reale și deci toate calculele aferente unui pas dublu **QR** pot fi efectuate în exclusivitate în aritmetică reală. Mai mult, matricea $\mathbf{H}_{k+2}$ din (124) fiind aceeași cu matricea H_{k+2} din (122) pasul dublu **QR** conservă și el structura superior Hessenberg. În concluzie, algoritmul **QR** cu deplasare explicită cu pas dublu are la bază următoarea schemă de calcul

1. Pentru $k = 1, 2, \dots$
 1. $s = h_{n-1, n-1} + h_{nn}$
 2. $p = h_{n-1, n-1} h_{nn} - h_{n, n-1} h_{n-1, n}$
 3. Se calculează $M = H^2 - sH + pI_n$
 4. Se factorizează $M = QR$
 5. $H \leftarrow Q^T H Q$

care, în forma implementabilă, se completează cu procedura de trunchiere constând în anularea efectivă a elementelor subdiagonale ce satisfac o condiție de tipul (118) sau (119) și monitorizarea elementelor subdiagonale nule pentru detectarea terminării (testul de terminare ar putea fi inexistența a două elemente subdiagonale nenule consecutive).

Schema asigură o convergență rapidă a matricei H către o structură cvasisuperior triunghiulară cu blocuri diagonale de dimensiune cel mult 2×2 care, fie că este chiar forma Schur reală urmărită, fie că permite calculul direct și imediat al acestei forme prin triangularizarea, cu ajutorul unor transformări de asemănare ortogonală, a blocurilor 2×2 cu valori proprii reale.

Motivul pentru care schema (127) nu constituie suportul implementărilor profesionale ale algoritmului **QR** este legat de slaba sa eficiență. Într-adevăr este ușor de văzut că un pas simplu **QR** are o complexitate $O(n^2)$, datorată în principal structurii Hessenberg cu care operează. Doi pași simpli **QR** vor avea aceeași complexitate. În schimb un pas dublu, în varianta (127) necesită calculul explicit al matricei M , crescând complexitatea la $O(n^3)$. Acest spor de efort de calcul nu are o fundamentare teoretică ci se datorează mai curând inabilității noastre de a organiza calculele în cadrul unui pas dublu **QR** în așa fel încât să păstrăm complexitatea $O(n^2)$. Așa cum se va vedea în secțiunea următoare, abilitatea necesară pentru evitarea creșterii complexității este departe de a fi banală.

9.6. Algoritmul QR cu deplasare implicită

Baza teoretică a elaborării unui algoritm eficient pentru un pas dublu **QR** este dată de teorema 6 conform căreia reducerea unei matrice la forma superior Hessenberg prin transformări ortogonale de asemănare are matricea ortogonală de transformare determinată, în esență, de prima sa coloană.

Concret, putem reduce complexitatea unui pas dublu **QR**, la $O(n^2)$ procedând în modul următor:

1. Se calculează prima coloană a matricei M , i.e.

$$m_1 = M(:,1) .$$
2. Se determină reflectorul elementar U_1 astfel ca

$$(U_1 m_1)(2:n) = 0 .$$

adică

$$U_1 m_1 = -\text{sign}(m_{11}) \|m_1\| e_1 = \rho e_1$$
3. Se calculează

$$N = U_1^T H_k U_1$$
4. Se aplică matricei N algoritmul **HQ** de aducere la forma superior Hessenberg:

$$\tilde{H} = Q^T N Q .$$

Procedura de mai sus calculează forma superior Hessenberg

$$\tilde{H} = (U_1 Q)^T H_k U_1 Q = \tilde{Q}^T H_k \tilde{Q}$$

cu matricea de transformare $\tilde{Q}$ având proprietatea

$$\tilde{Q}(:, 1) = U_1 Q e_1 = U_1 e_1 = \frac{1}{\rho} m_1 ,$$

penultima egalitate datorându-se faptului că $Q = \text{diag}(1, \overline{Q})$ (vezi algoritmul **HQ**), iar ultima rezultă din.

Pe de altă parte, din prima relație (124), datorită formei superior triunghiulare a matricei $\overline{R}_k$, rezultă

$$\overline{Q}_k e_1 = \frac{1}{r_{11}^{(k)}} \cdot m_1 .$$

În concluzie, matricele ortogonale de transformare $\tilde{Q}$ și $\overline{Q}$ au aceeași primă coloană și deci, conform teoremei 9.6, dacă matricele $\tilde{H}$ din (113) și H_{k+2} din (124) sunt ambele *irreductibile* atunci ele sunt, esențial, aceleași. Pentru a ne convinge de complexitatea $O(n^2)$ a procedurii vom explicita relațiile (128) - (132). Mai întâi vectorul m_1 are expresia

$$m_1 = \begin{bmatrix} h_{11}^2 + h_{12}h_{21} - sh_{11} + p \\ h_{21}(h_{11} + h_{22} - s) \\ h_{21}h_{32} \\ \dots \\ 0 \end{bmatrix}$$

în care am omis scrierea indicelui k asociat matricei H_k . Vectorul m_1 având numai trei componente nenule, calculul său necesită un număr de operații independent de n , deci are o complexitate $O(1)$. În plus, reflectorul U_1 din (129) are structura

$$U_1 = \text{diag}(\overline{U}_1, I_{n-3})$$

cu $U_1 \in R^{3 \times 3}$ a cărei determinare are, de asemenea, o complexitate $O(1)$.

În calculul matricei N sunt afectate numai primele trei linii și primele trei coloane ale matricei $H_k = H$. Într-adevăr, dacă partiționăm

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix}, \quad \mathbf{H}_{11} \in \mathbb{R}^{3 \times 3}$$

atunci

$$\mathbf{N} = \mathbf{U}_1^T \mathbf{H} \mathbf{U}_1 = \begin{bmatrix} \bar{\mathbf{U}}_1^T \mathbf{H}_{11} \bar{\mathbf{U}}_1 & \bar{\mathbf{U}}_1^T \mathbf{H}_{12} \\ \mathbf{H}_{21} \bar{\mathbf{U}}_1 & \mathbf{H}_{22} \end{bmatrix}.$$

În plus, blocul $\mathbf{H}_{21}$ are structura

$$\mathbf{H}_{21} = \begin{bmatrix} 0 & 0 & \mathbf{h}_{43} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$$

și, prin urmare,

$$\mathbf{H}_{21} \bar{\mathbf{U}}_1 = \begin{bmatrix} [0 \ 0 \ \mathbf{h}_{43}] \bar{\mathbf{U}}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \times & \times & \times \\ 0 & & \end{bmatrix} \begin{matrix} 1 \\ n-4 \end{matrix}$$

adică transformarea (131) alterează structura Hessenberg a matricei $\mathbf{H}$ numai în pozițiile (3,1), (4,1) și (4,2). Prin urmare, aplicarea algoritmului $\mathbf{H}\mathbf{Q}$ are ca obiectiv anularea, prin transformări ortogonale de asemănare, a celor trei elemente nenule ale matricei $\mathbf{N}$ alterante ale structurii superior Hessenberg. Pe de altă parte, această particularitate structurală (numai trei elemente în afara structurii Hessenberg în matricea inițială) obligă la adaptarea algoritmului $\mathbf{H}\mathbf{Q}$ astfel încât să se obțină o eficacitate maximă. Schema de calcul a acestei particularizări a algoritmului $\mathbf{H}\mathbf{Q}$ este următoarea.

1. Pentru $j = 1 : n-2$

1. Se determină reflectorul elementar U_{j+1} astfel încât

$$(U_{j+1} \mathbf{N})(i, j) = 0 \text{ pentru } i = j+2 : j+3 \text{ și } i \leq n.$$

2. $\mathbf{N} \leftarrow U_{j+1} \mathbf{N}$

3. $\mathbf{N} \leftarrow \mathbf{N} U_{j+1}$

cu precizarea că diferența față de algoritmul $\mathbf{H}\mathbf{Q}$ standard constă în faptul că de data aceasta, la fiecare pas se anulează numai două elemente (la ultimul, unul singur), celelalte elemente fiind nule din start.

SNu este greu de constatat că reflectorii elementari utilizați în schema (142) au structura

$$\begin{aligned} \mathbf{U}_{j+1} &= \text{diag}(\mathbf{I}_j, \bar{\mathbf{U}}_{j+1}, \mathbf{I}_{n-j-3}), \quad j = 1 : n-3 \\ \mathbf{U}_{n-1} &= \text{diag}(\mathbf{I}_{n-2}, \bar{\mathbf{U}}_{n-1}), \end{aligned}$$

unde $\bar{\mathbf{U}}_{j+1} \in \mathbb{R}^{3 \times 3}$, $\bar{\mathbf{U}}_{n-1} \in \mathbb{R}^{2 \times 2}$ sunt reflectorii elementari de ordinul 3, respectiv 2.

Pe această bază rezultă imediat că numărul asimptotic de operații aritmetice necesar pentru execuția schemei (142) este $\mathbf{N}_{\text{op}} \cong 10n^2$, la care se adaugă $n-2$ extrageri de radicali. Dacă se dorește acumularea transformărilor, respectiv calculul matricei $\mathbf{Q}$ din relația (132)

$$\mathbf{Q} = \mathbf{U}_1 \mathbf{U}_2 \dots \mathbf{U}_{n-1},$$

sunt necesare $\mathbf{N}'_{\text{op}} \cong 10n^2$ operații suplimentare.

Exemplificăm, pentru clarificare, evoluția structurală datorată execuției schemei (142), pentru cazul particular $n = 5$.

$$\begin{aligned}
 N &= \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ \otimes & \otimes & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} \Rightarrow \\
 N \leftarrow U_2 N &= \begin{bmatrix} \times & \times & \times & \times & \times \\ \bar{\times} & \bar{\times} & \bar{\times} & \bar{\times} & \bar{\times} \\ \emptyset & \times & \times & \times & \times \\ \emptyset & \otimes & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} \Rightarrow N \leftarrow NU_2 = \begin{bmatrix} \times & \bar{\times} & \bar{\times} & \bar{\times} & \times \\ \times & \bar{\times} & \times & \times & \times \\ 0 & \bar{\times} & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \\ 0 & \otimes & \otimes & \times & \times \end{bmatrix} \Rightarrow N \leftarrow U_3 N = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \bar{\times} & \bar{\times} & \bar{\times} & \bar{\times} \\ 0 & \emptyset & \times & \times & \times \\ 0 & \emptyset & \otimes & \times & \times \end{bmatrix} \Rightarrow \\
 N \leftarrow NU_3 &= \begin{bmatrix} \times & \times & \bar{\times} & \bar{\times} & \bar{\times} \\ \times & \times & \bar{\times} & \bar{\times} & \bar{\times} \\ 0 & \times & \bar{\times} & \bar{\times} & \bar{\times} \\ 0 & 0 & \bar{\times} & \bar{\times} & \bar{\times} \\ 0 & 0 & \otimes & \times & \bar{\times} \end{bmatrix} \Rightarrow N \leftarrow U_4 N = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \bar{\times} & \bar{\times} & \bar{\times} \\ 0 & 0 & \emptyset & \times & \bar{\times} \end{bmatrix} \Rightarrow N \leftarrow NU_4 = \begin{bmatrix} \times & \times & \times & \bar{\times} & \bar{\times} \\ \times & \times & \times & \bar{\times} & \bar{\times} \\ 0 & \times & \times & \bar{\times} & \bar{\times} \\ 0 & 0 & \times & \bar{\times} & \bar{\times} \\ 0 & 0 & 0 & \bar{\times} & \bar{\times} \end{bmatrix}.
 \end{aligned}$$

În diagramele de mai sus simbolul $\otimes$ marchează elementele care alterează structura Hessenberg, simbolul $\emptyset$ marchează elementele anulate la pasul curent, iar încadrările marchează elementele afectate de multiplicarea matriceală curentă.

În concluzie, complexitatea pasului dublu **QR** cu deplasare implicită, definit de (128)-(132) și prezentat prima dată de J. Francis [9] și V.N. Kublanovskaia [20] în 1961 este $O(n^2)$, această formă reprezentând baza implementărilor performante ale algoritmului **QR** de reducere iterativă a unei matrice la forma Schur reală.

În scrierea efectivă a algoritmului de implementare a unui pas dublu **QR** cu deplasare implicită, din motive de concizie, vom utiliza funcțiile **refmat** și **matref** de premultiplicare, respectiv de postmultiplicare cu un reflector elementar (vezi algoritmii 9.3 și 9.4), al căror apel ține seama de structura specială din acest caz. De asemenea, nu se realizează calculul efectiv al matricei de transformare dar vor fi furnizate elementele definitorii ale reflectorilor utilizați. Concret, pentru calculul matricei Q este suficientă cunoașterea reflectorilor de ordinul 3 și 2

$$\begin{aligned}
 \bar{U}_j &= I_3 - \frac{\mathbf{u}_j \mathbf{u}_j^T}{\beta_j}, \quad \mathbf{u}_j \in \mathbb{R}^3, \quad j = 1 : n - 2 \\
 \bar{U}_{n-1} &= I_2 - \frac{\mathbf{u}_{n-1} \mathbf{u}_{n-1}^T}{\beta_{n-1}}, \quad \mathbf{u}_{n-1} \in \mathbb{R}^2,
 \end{aligned}$$

ale căror elemente definitorii vor fi memorate într-o matrice $\mathbf{V} \in \mathbb{R}^{3 \times (n-1)}$ astfel încât

$$\mathbf{V}(:, j) = \mathbf{u}_j, \quad j = 1 : n - 2, \quad \mathbf{V}(:, n - 1) = \begin{bmatrix} \mathbf{u}_{n-1} \\ 0 \end{bmatrix} \text{ și un vector } \mathbf{b} \in \mathbb{R}^{n-1} \text{ al}$$

scalarilor $\beta_j, \quad j = 1 : n - 1$.

Cu aceste precizări, prezentăm mai jos algoritmul de implementare a unui pas dublu **QR** cu deplasare implicită, cu comentarii care să permită identificarea simplă a considerațiilor teoretice de mai sus.

Algoritmul 6

```

[H,V,b] = QR2(n,H)
/* Implementarea unui pas dublu QR cu deplasare implicită.
/* Intrări:  n = ordinul matricei Hessenberg ireductibile de intrare;
/*          H = matricea reală superior Hessenberg ireductibilă
/* Ieșiri:   H ← QTHQ - suprascrierea matricei de intrare cu matricea
/*          succesori din șirul QR cu pași dubli;
/*          V ∈ R3×(n-1) matrice cu elementele vectorilor de ordinul 3 ce
/*          definesc reflectorii utilizați;
/*          b ∈ Rn-1 vectorul ce conține scalarii βj.

1. /* Calculul sumei și produsului deplasărilor (126)
    1. s = H(n-1,n-1) + H(n,n)
    2. p = H(n-1,n-1) · H(n,n) - H(n-1,n) · H(n,n-1)
2. /* Calculul elementelor nenule din m1 cu (136)
    1. m1(1) = (H(1,1))2 + H(1,2) · H(2,1) - sH(1,1) + p
    2. m1(2) = H(2,1) · (H(1,1) + H(2,2) - s)
    3. m1(3) = H(2,1) · H(3,2)
3. /* Calculul reflectorului elementar U1 din (137)
    1. σ = sign(m1(1)) · ||m1||2
    2. u(1) = m1(1) + σ
    3. u(2:3) = m1(2:3)
    4. V(:,1) = u
    5. b(1) ← β1 = u(1) · σ
4. /* Calculul matricei H ← N = U1THU1 din (131)
    1. [H(1:3,:)] = refmat(u,b(1),H(1:3,:))
    2. [H(1:4,1:3)] = matref(H(1:4,1:3),u,b(1))
5. /* Reducerea lui H la forma superior Hessenberg
    1. Pentru j = 1 : n-3
        1. /* Calculul reflectorului Uj+1
            1. σ = sign(H(j+1,j)) · ||H(j+1:j+3,j)||2
            2. u(1) = H(j+1,j) + σ
            3. u(2:3) = H(j+2:j+3,j)
            4. V(:,j+1) = u
            5. b(j+1) ← βj+1 = u(1) · σ
        2. /* Calcul H ← Uj+1THUj+1
            1. H(j+1,j) = -σ
            2. H(j+2:j+3,j) = 0
            3. [H(j+1:j+3,j+1:n)] =
                refmat(u,b(j+1),H(j+1:j+3,j+1:n))
            4. r = min(j+4,n)
            5. [H(1:r,j+1:j+3)] =

```

```

matref( $H(1:r, j+1:j+3), u, b(j+1)$ )
2. /* Calculul reflectorului  $U_{n-1}$ 
   1.  $\sigma = \text{sign}(H(n-1, n-2)) \cdot \|H(n-1:n, n-2)\|_2$ 
   2.  $u(1) = H(n-1, n-2) + \sigma$ 
   3.  $u(2) = H(n, n-2)$ 
   4.  $u(3) = 0$ 
   5.  $V(:, n-1) = u$ 
   6.  $b(n-1) \leftarrow \beta_{n-1} = u(1) \cdot \sigma$ 
3. /* Calculul  $H \leftarrow U_{n-1}^T H U_{n-1}$ 
   1.  $H(n-1, n-2) = -\sigma$ 
   2.  $H(n, n-2) = 0$ 
   3.  $[H(n-1:n, n-1:n)] =$ 
       refmat( $u(1:2), b(n-1), H(n-1:n, n-1:n)$ )
   4.  $[H(1:n, n-1:n)] =$ 
       matref( $H(1:n, n-1:n), u(1:2), b(n-1)$ )

```

Algoritmul 5 de reducere la formă superior Hessenberg și algoritmul 9.6 de iterare în construcția șirului **QR** reprezintă baza programelor profesionale de calcul a valorilor proprii (cum ar fi cele din sistemele de programe științifice EISPACK, LAPACK, MATLAB, MATEMATICA).

Pentru a descrie întregul proces de calcul, cunoscut sub numele de algoritmul **QR**, trebuie să controlăm procesul iterativ prin anularea efectivă a elementelor subdiagonale la scăderea lor sub un prag precizat, monitorizarea elementelor subdiagonale nule și, pe această cale, creșterea eficienței prin reducerea progresivă a dimensiunii.

Pentru implementarea acestor aspecte vom utiliza faptul că deflația iterativă are loc, cel mai rapid, în raport cu colțul din dreapta jos al matricei H și, cu o viteză de convergență ceva mai redusă, în raport cu colțul din stânga sus. În consecință, se impune, în monitorizarea zerourilor subdiagonale, partiția, la fiecare iterație, a matricei curente H din șirul **QR** în forma

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ 0 & H_{22} & H_{23} \\ 0 & 0 & H_{33} \end{bmatrix}, \quad H_{11} \in \mathbb{R}^{p \times p}, \quad H_{33} \in \mathbb{R}^{q \times q},$$

astfel încât blocul diagonal H_{33} să aibă o structură cvasisuperior-triunghiulară (adică o structură cu blocuri diagonale de dimensiuni cel mult 2×2) de dimensiune maximă, iar blocul H_{22} să fie superior Hessenberg *irreductibil*, de asemenea de dimensiune maximă (ceea ce implică p minim). Cu această partiție iterația curentă se materializează în aplicarea pasului dublu **QR** submatricei H_{22}

$$H_{22} \leftarrow Q_{22}^T H_{22} Q_{22},$$

ceea ce derivă din aplicarea transformării ortogonale de asemănare globale

$$H \leftarrow \text{diag}(I_p, Q_{22}^T, I_q) \cdot H \cdot \text{diag}(I_p, Q_{22}, I_q),$$

cu următoarele efecte asupra celorlalte blocuri ale partiției

$$H_{12} \leftarrow H_{12} \cdot Q_{22}, \quad H_{23} \leftarrow Q_{22}^T \cdot H_{23}.$$

Algoritmul își epuizează funcția în momentul în care întreaga matrice H devine cvasisuperior triunghiulară, respectiv parametrul q atinge dimensiunea n a matricei.

Rezultatul procesului iterativ, cu monitorizarea structurală de mai sus, este o structură a matricei H având blocuri diagonale de dimensiune 1×1 sau 2×2 . Pentru a obține forma Schur reală, care este obiectivul declarat al algoritmului **QR**, trebuie triangularizate blocurile 2×2 cu valori proprii reale. Pentru a vedea cum se poate realiza această operație, vom proceda la deflația (vezi 9.2) unei matrice $B \in R^{2 \times 2}$ cu

$$b_{21} \neq 0$$

și satisfăcând condiția

$$\Delta = (b_{11} - b_{22})^2 + 4b_{12}b_{21} \geq 0,$$

de existență a valorilor proprii reale

$$\lambda_1 = \frac{b_{11} + b_{22} - \sqrt{\Delta}}{2}, \quad \lambda_2 = \frac{b_{11} + b_{22} + \sqrt{\Delta}}{2}.$$

Determinând un vector propriu unitar asociat lui λ_1 și completându-l până la o matrice ortogonală, este ușor de văzut că transformarea ortogonală de asemănare definită de matricea de rotație

$$P = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

cu

$$c = \frac{\lambda_1 - b_{22}}{\sqrt{b_{21}^2 + (\lambda_1 - b_{22})^2}}, \quad s = \frac{-b_{21}}{\sqrt{b_{21}^2 + (\lambda_1 - b_{22})^2}}$$

conduce la realizarea triangularizării urmărite

$$C = P^T B P = \begin{bmatrix} \lambda_1 & b_{12} - b_{21} \\ 0 & \lambda_2 \end{bmatrix}.$$

La nivelul matricei cvasisuperior triunghiulare H , dacă blocul diagonal 2×2 cu valori proprii reale se află pe liniile și coloanele $k, k + 1$, atunci pentru triangularizarea lui se aplică o transformare de forma

$$H \leftarrow \text{diag} (I_{k-1}, P^T, I_{n-k-1}) \cdot H \cdot \text{diag} (I_{k-1}, P, I_{n-k-1})$$

cu P calculat ca mai sus.

La fel ca și în cazul reflectorilor, pentru descrierea algoritmică a transformării (156) vom introduce două proceduri. Prima procedură, numită **rotmat**, va realiza premultiplicarea, cu suprascrisere, $A \leftarrow PA$ a unei matrice $A \in R^{2 \times p}$ cu o rotație $P \in R^{2 \times 2}$ având expresia (153) și dată de parametri scalari c și s .

Algoritmul 7.

$[A] = \text{rotmat}(c, s, A)$

/* Premultiplicarea unei matrice cu o rotație.

/* Intrări: c, s = parametri definitori ai rotației;

/* $A \in R^{2 \times p}$ = matricea dată.

/* Ieșiri: $A \leftarrow PA$.

1. Pentru $j = 1 : p$

1. $\alpha = cA(1, j) + sA(2, j)$

2. $A(2, j) \leftarrow -sA(1, j) + cA(2, j)$

$$3. A(1, j) = \alpha$$

Evident, un apel $[A] = \text{rotmat}(c, -s, A)$ va realiza suprascrierea $A \leftarrow P^T A$.

A doua procedură, numită **matrot**, va realiza postmultiplicarea, cu suprascriere, $A \leftarrow AP$ a unei matrice $A \in R^{m \times 2}$ cu o rotație $P \in R^{2 \times 2}$ având expresia (153) și dată de parametri scalari c și s .

Algoritmu 8.

$[A] = \text{matrot}(A, c, s)$

/* Postmultiplicarea unei matrice cu o rotație.

/* Intrări: c, s = parametri definitori ai rotației;

/* $A \in R^{m \times 2}$ = matricea dată.

/* Ieșiri: $A \leftarrow AP$.

1. Pentru $i = 1 : m$

1. $\alpha = A(i, 1) \cdot c - A(i, 2) \cdot s$

2. $A(i, 2) \leftarrow A(i, 1) \cdot s + A(i, 2) \cdot c$

3. $A(i, 1) = \alpha$.

Suntem în măsură acum să prezentăm, în toată plenitudinea sa, algoritmul **QR**. Toate matricele șirului **QR** vor suprascrie matricea inițială dată. În mod facultativ, pe baza variabilei de opțiune **opt** algoritmul calculează matricea de transformare ortogonală cumulată Q . Pentru concizie, produsele matriceale nu au fost explicitate, dar se atrage atenția că ele trebuie făcute cât mai economicos, exploatând particularitățile structurale ale matricelor factor.

Algoritmul 9.

$[A, Q] = \text{QR}(n, A, \varepsilon, \text{opt})$

/* Algoritmul **QR** de reducere a unei matrice reale la forma Schur

/* reală prin transformări ortogonale de asemănare.

/* Intrări: n = ordinul matricei inițiale A ;

/* A = matricea dată;

/* ε = toleranța pentru detectarea elementelor de modul neglijabil;

/* **opt** = opțiunea de calcul a matricei de transformare ('Y' sau 'N').

/* Ieșiri: $A \leftarrow S = Q^T A Q$ - matricea redusă la forma Schur reală;

/* Q = matricea ortogonală de transformare.

1. /* Inițializarea matricei Q

1. $Q = I_n$

2. /* Reducerea matricei A la formă superior Hessenberg

1. $[A, Q, V, b] = \text{HQ}(n, A, \text{opt})$

3. /* Reducerea matricei superior Hessenberg la formă cvasisuperior triunghiulară

1. $p = 0$

2. $q = 0$

3. Cât timp $q < n$

1. /* Anularea elementelor subdiagonale neglijabile

1. Pentru $i = p + 1 : n - q - 1$

1. Dacă $|A(i + 1, i)| \leq \varepsilon \cdot (|A(i, i)| + |A(i + 1, i)|)$ atunci

```

1.  $A(i+1,i) = 0$ 
2./* Stabilirea valorii maxime pentru  $q$  și a valorii minime
   pentru  $p$  astfel încât blocul superior Hessenberg  $A_{22}$  din
   (122) să fie ireductibil de dimensiune maximă.
1. test = TRUE
2. Cât timp test = TRUE și  $q < n$ 
   1. Dacă  $A(n-q, n-q-1) = 0$  atunci
       1.  $q \leftarrow q+1$ 
   altfel
       1. Dacă  $q < n-1$  și  $A(n-q-1, n-q-2) = 0$  atunci
           1.  $q \leftarrow q+2$ 
       altfel
           1. test = FALSE
3. Dacă  $q < n-2$  atunci
   1.  $p = n-q-1$ 
   2. Cât timp  $A(p, p-1) \neq 0$ 
       1.  $p \leftarrow p-1$ 
   altfel
   1. /* Matricea A are formă cvasisuperior triunghiulară.
      /*Triangularizarea blocurilor diagonale  $2 \times 2$  cu valori
      proprii reale
   1.  $i = 1$ 
   2. Cât timp  $i < n$ 
       1. Dacă  $A(i+1,i) \neq 0$  atunci
           1.  $\delta = (A(i,i) - A(i+1,i+1))^2 + 4A(i+1,i)A(i,i+1)$ 
           2. Dacă  $\delta \geq 0$  atunci
               1.  $\sigma = A(i,i) + A(i+1,i+1)$ 
               2.  $\delta \leftarrow \sqrt{\delta}$ 
               3.  $\lambda_1 = \frac{\sigma - \delta}{2}$ 
               4.  $\lambda_2 = \frac{\sigma + \delta}{2}$ 
               5.  $\rho = \sqrt{(A(i+1,i))^2 + (\lambda_1 - A(i+1,i+1))^2}$ 
               6.  $c = \frac{\lambda_1 - A(i+1,i+1)}{\rho}$ ,  $s = \frac{-A(i+1,i)}{\rho}$ 
               7.  $A(i,i) = \lambda_1$ ,  $A(i+1,i+1) = \lambda_2$ 
               8.  $A(i,i+1) \leftarrow A(i,i+1) - A(i+1,i)$ 
               9.  $A(i+1,i) = 0$ 
           10. Dacă  $i > 1$  atunci
               1.  $A(1:i-1, i:i+1) =$ 
                  matrot( $A(1:i-1, i:i+1), c, s$ )
           11. Dacă  $i+1 < n$  atunci
               1.  $A(i:i+1, i+1:n) =$ 
                  rotmat( $c, -s, A(i:i+1, i+1:n)$ )

```

```

12. Dacă  $\text{opt} = 'Y'$  atunci
    1.  $Q(:, i:i+1) = \text{matrot}(Q(:, i:i+1), c, s)$ 
13.  $i \leftarrow i+2$ 
altfel
    1.  $i \leftarrow i+1$ 
3. /* Terminarea algoritmului
    1. RETURN
4. /* Aplicarea unui pas dublu QR cu deplasare implicită blocului diagonal  $A_{22}$ 
    1.  $[A(p+1:n-q, p+1:n-q), V, b] =$ 
        QR2( $n-p-q, A(p+1:n-q, p+1:n-q)$ )
    2. Pentru  $i = 1:n-p-q-2$ 
        1.  $v = V(1:3, i)$ 
        2.  $m = p+i$ 
        3. Dacă  $p > 0$  atunci
            1.  $A(1:p, m:m+2) = \text{matref}(A(1:p, m:m+2), v, b(i))$ 
        4. Dacă  $q > 0$  atunci
            1.  $A(m:m+2, n-q+1:n) =$ 
                refmat( $v, b(i), A(m:m+2, n-q+1:n)$ )
        5. Dacă  $\text{opt} = 'Y'$  atunci
            1.  $Q(:, m:m+2) = \text{matref}(Q(:, m:m+2), v, b(i))$ 
    3. /* Idem pentru  $i = n-p-q-1$ 
        1.  $m = n-q-1$ 
        2.  $v = V(1:2, n-p-q-1)$ 
        3. Dacă  $p > 0$  atunci
            1.  $A(1:p, m:m+1) =$ 
                matref( $A(1:p, m:m+1), v, b(i)$ )
        4. Dacă  $q > 0$  atunci
            1.  $A(m:m+1, m:m+1) =$ 
                refmat( $v, b(i), A(m:m+1, m:m+1)$ )
        5. Dacă  $\text{opt} = 'Y'$  atunci
            1.  $Q(:, m:m+1) =$ 
                matref( $Q(:, m:m+1), v, b(n-p-q-1)$ )

```

Evaluarea complexității algoritmului **QR** nu poate fi făcută decât în mod empiric, datorită prezenței procesului iterativ. Statisticile și numeroasele experimente numerice arată că, în medie, sunt suficiente două iterații **QR** pentru anularea convențională a unui element subdiagonal. În această ipoteză, pentru calculul formei Schur reale sunt necesare $\mathbf{N}'_{\text{op}} \cong 10n^3$ operații aritmetice, iar pentru calculul matricei de transformare, alte $\mathbf{N}''_{\text{op}} \cong 15n^3$ operații, ceea ce încadrează algoritmul **QR** în categoria algoritmilor cu complexitate $\mathbf{O}(n^3)$.

Observații. Așa cum s-a prezentat în secțiunea 9.2 forma Schur reală a unei matrice este determinată până la o ordonare a blocurilor diagonale. Forma Schur reală calculată de algoritmul 9.9 nu satisface nici un criteriu, a priori fixat, privitor la ordinea blocurilor diagonale.

În momentul în care o formă Schur a unei matrice date satisface un criteriu precizat de ordine a blocurilor diagonale, este numită în mod uzual, *formă Schur reală ordonată*. Pentru a putea

ordona după dorință blocurile diagonale este suficient să dispunem de un mijloc de permutare a două elemente (blocuri) diagonale vecine. În cazul nostru, un astfel de instrument trebuie să fie o transformare de asemănare ortogonală. Propunem cititorului ca exercițiu, găsirea transformărilor ortogonale de asemănare care permută două blocuri diagonale vecine în toate situațiile structurale posibile, respectiv: a) două blocuri scalare; b) două blocuri 2×2 ; c) un bloc 1×1 cu un bloc 2×2 și d) un bloc 2×2 cu un bloc 1×1 . Dacă apar dificultăți consultați, de exemplu, lucrarea [5].

Ordonarea formei Schur (reale) permite evidențierea unor subspații invariante asociate unui grup de valori proprii și calculul unor baze ortogonale pentru acestea. Într-adevăr, dacă $S = Q^T A Q$ este forma Schur reală ordonată astfel încât în partiția

$$S = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}, \quad S_{11} \in \mathbb{R}^{p \times p},$$

blocul S_{11} conține valorile proprii de interes, atunci este ușor de văzut că subspațiul liniar $S_1 = \text{Im } Q_1$, unde $Q_1 = Q(:, 1 : p)$, este un subspațiu A -invariant, asociat grupului de valori proprii $\lambda(S_{11}) \subset \lambda(A)$, iar coloanele matricei Q_1 formează o bază ortogonală a acestui subspațiu. De aceea, în [29] este folosit pe scară largă conceptul de *sistem propriu* al unei matrice, definit de o pereche (grup de valori proprii, subspațiu invariant asociat), ca o generalizare naturală a conceptului (valoare proprie, vector (direcție) proprie asociat(ă)).

Calculul vectorilor proprii

Algoritmul 9 calculează forma Schur reală $S = Q^T A Q$ a unei matrice $A \in \mathbb{R}^{n \times n}$ date. Conform cu cele prezentate în secțiunea 9.1, dacă y_k este un vector propriu al matricei S , asociat valorii proprii $\lambda_k \in \lambda(S) = \lambda(A)$, respectiv

$$S y_k = \lambda_k y_k,$$

atunci

$$x_k = Q y_k$$

este un vector propriu al matricei A asociat aceleași valori proprii λ_k . În consecință, dacă algoritmul QR calculează matricea de transformare Q , calculul vectorilor proprii asociați valorilor proprii reale se reduce la rezolvarea sistemelor liniare (166) și multiplicărilor (167). Dacă λ_k este o valoare proprie distinctă, rezolvarea sistemului omogen (166) nu ridică probleme. În acest caz sistemul (166) are forma

$$\begin{bmatrix} S_{11} - \lambda_k I_{k-1} & S_{12} & S_{13} \\ 0 & 0 & S_{23} \\ 0 & 0 & S_{33} - \lambda_k I_{n-k} \end{bmatrix} \cdot \begin{bmatrix} y'_k \\ y''_k \\ y'''_k \end{bmatrix} = 0$$

cu $y''_k = y_k(k)$ și cu blocurile $S_{11} - \lambda_k I_{k-1}$ și $S_{33} - \lambda_k I_{n-k}$ nesarabile. De aici rezultă $y'''_k = 0$ și deci vectorul propriu are structura

$$y_k = \alpha \begin{bmatrix} - (S_{11} - \lambda_k I_{k-1})^{-1} S_{12} \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \} k-1 \\ \} 1 \\ \} n-k \end{matrix}$$

cu $\alpha \in \mathbb{R} \setminus \{0\}$ arbitrar. În concluzie, calculul lui y_k se reduce la rezolvarea unui sistem liniar cvasitriunghiular de ordinul $k-1$. În cazul valorilor proprii reale multiple (identitatea a două valori proprii calculate este practic imposibilă datorită aritmeticii aproximative) apar

dificultăți importante datorită proastei condiționări numerice a matricelor $\mathbf{S}_{11} - \lambda_k \mathbf{I}_{k-1}$ și/sau $\mathbf{S}_{33} - \lambda_k \mathbf{I}_{n-k}$ și de aceea se recomandă utilizarea unor procedee speciale.

Pentru valori proprii complex conjugate, distincte,

$$\lambda_{k,k+1} = \alpha_k \pm i\beta_k = \lambda(\mathbf{S}_{22}) \subset \lambda(\mathbf{S}) = \lambda(\mathbf{A})$$

vectorii proprii asociați matricei $\mathbf{S}$ pot avea forma

$$\mathbf{y}_{k,k+1} = \mathbf{u}_k \pm i\mathbf{v}_k, \quad \mathbf{u}_k, \mathbf{v}_k \in \mathbb{R}^n.$$

Vectorii reali u_k, v_k se pot calcula prin rezolvarea sistemului omogen singular $2n$ dimensional

$$\begin{cases} \mathbf{S}\mathbf{u}_k = \alpha_k \mathbf{u}_k - \beta_k \mathbf{v}_k \\ \mathbf{S}\mathbf{v}_k = \beta_k \mathbf{u}_k + \alpha_k \mathbf{v}_k \end{cases}$$

Cum $\beta_k \neq 0$

$$\begin{cases} \mathbf{u}_k = \frac{1}{\beta_k} (\mathbf{S} - \alpha_k \mathbf{I}_n) \mathbf{v}_k \\ (\mathbf{S}^2 - 2\alpha_k \mathbf{S} + (\alpha_k^2 + \beta_k^2) \mathbf{I}_n) \mathbf{v}_k = 0. \end{cases}$$

Pe baza partiției

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} & \mathbf{S}_{13} \\ 0 & \mathbf{S}_{22} & \mathbf{S}_{23} \\ 0 & 0 & \mathbf{S}_{33} \end{bmatrix}, \quad \mathbf{S}_{11} \in \mathbb{R}^{(k-1) \times (k-1)},$$

$$\mathbf{S}_{22} \in \mathbb{R}^{2 \times 2}, \quad \mathbf{S}_{33} \in \mathbb{R}^{(n-k-1) \times (n-k-1)}$$

acceptate deja, a matricei $\mathbf{S}$, a doua ecuație se scrie

$$\begin{bmatrix} \mathbf{S}_{11}^2 - 2\alpha_k \mathbf{S}_{11} + (\alpha_k^2 + \beta_k^2) \mathbf{I}_{k-1} & \mathbf{S}_{11}\mathbf{S}_{12} + \mathbf{S}_{12}\mathbf{S}_{22} & \times \\ 0 & 0 & \times \\ 0 & 0 & \mathbf{S}_{33}^2 - 2\alpha_k \mathbf{S}_{33} + (\alpha_k^2 + \beta_k^2) \mathbf{I}_{n-k} \end{bmatrix} \begin{bmatrix} \mathbf{v}'_k \\ \mathbf{v}''_k \\ \mathbf{v}'''_k \end{bmatrix} = 0$$

unde s-a ținut seama de faptul că blocul $\mathbf{S}_{22}$ este anulador pentru propriul polinom caracteristic (Cayley-Hamilton), iar $\times$ marchează blocurile neinteresante.

Dacă valorile proprii sunt distincte de celelalte valori proprii ale matricei $\mathbf{S}$, atunci matricele

$$\mathbf{T}_{11} = \mathbf{S}_{11}^2 - 2\alpha_k \mathbf{S}_{11} + (\alpha_k^2 + \beta_k^2) \mathbf{I}_{k-1} \quad \text{și}$$

$$\mathbf{T}_{33} = \mathbf{S}_{33}^2 - 2\alpha_k \mathbf{S}_{33} + (\alpha_k^2 + \beta_k^2) \mathbf{I}_{n-k-1}$$

sunt nesingulare și, prin urmare, ecuația (175) are ca soluție un vector $\mathbf{v}_k$ având

$$\begin{cases} \mathbf{v}'''_k = 0 \\ \mathbf{v}''_k = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \neq 0 \quad \text{arbitrar} \\ \mathbf{v}'_k = -\mathbf{T}_{11}^{-1}(\mathbf{S}_{11}\mathbf{S}_{12} + \mathbf{S}_{12}\mathbf{S}_{22})\mathbf{v}''_k \end{cases}$$

$\mathbf{u}_k$ obținându-se din prima relație. Vectorii proprii ai matricei inițiale $\mathbf{A}$ asociate aceluiași valori proprii și având structura

$$\mathbf{x}_{k,k+1} = \mathbf{z}_k \pm i\mathbf{w}_k, \quad \mathbf{z}_k, \mathbf{w}_k \in \mathbb{R}^n$$

se calculează separat părțile reală și imaginară

$$\begin{cases} \mathbf{z}_k = \mathbf{Q}\mathbf{u}_k \\ \mathbf{w}_k = \mathbf{Q}\mathbf{v}_k \end{cases}.$$

O alternativă pentru tehnicile de calcul de mai sus a vectorilor proprii, recomandată mai ales în cazurile în care se cere un set redus de vectori proprii, evită acumularea transformărilor în matricea $\mathbf{Q}$ și apelează la metoda puterii inverse conform schemei:

1. Reducerea matricei la forma superior Hessenberg cu algoritmul $\mathbf{H}\mathbf{Q}$

$$\mathbf{A} \leftarrow \mathbf{H} = \mathbf{U}^T \mathbf{A} \mathbf{U}.$$

2. Se aplică iterarea $\mathbf{QR}$ cu pas dublu cu deplasare implicită, fără acumularea transformărilor până la evidențierea valorii proprii λ , al cărei vector propriu asociat dorim să-l calculăm.

3. Se aplică algoritmul 9.2 (metoda puterii inverse) matricei $\mathbf{H}$ cu deplasarea inițială $\mu = \lambda$ care produce un vector $\mathbf{y}$ astfel încât $\mathbf{H}\mathbf{y} = \lambda\mathbf{y}$.

4. Se calculează vectorul propriu căutat $\mathbf{x} = \mathbf{U}\mathbf{y}$.

Dacă se doresc mai mulți vectori proprii, se repetă pașii 2-4 de mai sus. Iterarea inversă pentru o matrice superior Hessenberg reprezintă o soluție foarte economică întrucât practic s-a constatat că este suficientă o singură iterație (de complexitate $\mathcal{O}(n^2)$), pentru calculul vectorului propriu $\mathbf{y}$. În acest fel, partea cea mai complexă a schemei de mai sus este reducerea la forma superior Hessenberg și evidențierea iterativă a valorii proprii dorite. Dacă numărul vectorilor proprii doriți este mare, (de exemplu mai mare decât $\frac{n}{4}$) atunci iterarea $\mathbf{QR}$ devine comparabilă cu cea

din algoritmul 9.9 și avantajul schemei de mai sus nu mai este evident.

În încheiere dorim să menționăm din nou faptul că în multe aplicații vectorii proprii pot fi înlocuiți cu succes cu vectorii Schur, respectiv de coloanele matricei de transformare $\mathbf{Q}$, al căror calcul opțional este luat în considerare de către algoritmul 9.9 și se efectuează cu o mare acuratețe.