CURS 4

METODE NUMERICE PENTRU PROBLEMA DE VALORI PROPRII

Partea II

Algoritmul QR

1 Algoritmul QR

1.1 Metoda

QR este unul din cei mai utilizaţi algoritmi pentru a determina toate valorile proprii ale unei matrici. În numele metodei, *Q* desemnează o matrice ortogonală, iar *R* o matrice superior-triunghiulară (*R* vine de la *right*).

Considerăm o *matrice reală* nesingulară **A**, simetrică sau nu.

Esenţa algoritmului *QR* constă în următoarea proprietate:

> Dacă **A** este factorizată în produsul $\mathbf{A} = \mathbf{QR}$, unde **Q** este nesingulară, atunci matricea produsului în ordine inversă, $\mathbf{A}' = \mathbf{RQ}$, are aceleaşi valori proprii ca şi **A** (fiind similară cu **A)** ■
>
> Într-adevăr, avem: $\mathbf{R} = \mathbf{Q}^{-1}\mathbf{A}$, şi $\mathbf{A}' = \mathbf{Q}^{-1}\mathbf{AQ}$.

Algoritmul *QR* realizează factorizări succesive ale şirului de matrici $\{ \mathbf{A}_k \}$, unde $\mathbf{A}_1 = \mathbf{A}$, definite de:

$\mathbf{A}_1 = \mathbf{Q}_1\mathbf{R}_1$; $\quad\quad$ $\mathbf{A}_2 = \mathbf{R}_1\mathbf{Q}_1$;

$\mathbf{A}_2 = \mathbf{Q}_2\mathbf{R}_2$; $\quad\quad$ $\mathbf{A}_3 = \mathbf{R}_2\mathbf{Q}_2$;

…

$\mathbf{A}_k = \mathbf{Q}_k\mathbf{R}_k$; $\quad\quad$ $\mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k$;

…

Matricile $\mathbf{A}_k$, şi $\mathbf{Q}_k, \mathbf{R}_k$, au următoarele proprietăţi:

1) Toate $\mathbf{A}_k$ au aceleaşi valori proprii ca şi **A**.

2) Dacă **A** este *simetrică*, sau *tridiagonală*, sau **A** are forma *Hessenberg superioară*, matricile $\mathbf{A}_k$ *vor păstra aceste forme*.

3) Fie $\overline{\mathbf{Q}}_k = \mathbf{Q}_1\mathbf{Q}_2 \ldots \mathbf{Q}_k$, atunci (prin inducţie): $\mathbf{A}_{k+1} = \overline{\mathbf{Q}}_k{}^{-1}\mathbf{A}_1\overline{\mathbf{Q}}_k$.

4) Fie $\overline{\mathbf{R}}_k = \mathbf{R}_k \ldots \mathbf{R}_1$, atunci: $\overline{\mathbf{Q}}_k\overline{\mathbf{R}}_k = (\mathbf{A}_1)^k$

Matrici Hessenberg şi Househooder

Forma **Hessenberg** (superioară) este o matrice avînd elementele zero sub sub-diagonală. De exemplu, o matrice Hessenberg 6×6 arată astfel:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \ldots & & & & a_{16} \\ a_{21} & a_{22} & \ldots & & & \vdots \\ 0 & a_{32} & a_{33} & \ldots & & \\ 0 & 0 & a_{43} & a_{44} & \ldots & \\ 0 & 0 & 0 & a_{54} & a_{55} & \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

■

O matrice **Householder** *n*×*n* este definită prin

$$\mathbf{P} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^T,$$

unde **I** este matricea unitate *n*×*n*, iar **w** este un vector cu *n* coordonate, normalizat relativ la norma euclidiană, adică $\| \mathbf{w} \|_2 = 1$, sau $\mathbf{w}^T\mathbf{w} = 1$, şi altfel arbitrar.

Explicit,

$$\mathbf{P} = \begin{bmatrix} 1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 1 \end{bmatrix} - 2 \begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_n \end{bmatrix} \begin{bmatrix} w_1 & \ldots & w_j & \ldots & w_n \end{bmatrix}$$

$$p_{ij} = \delta_{ij} - 2w_i w_j$$

Matricea **P** este *simetrică* şi *este propria sa inversă* (este unitară), întrucât avem

$\mathbf{P}^T = \mathbf{P}$, şi $\mathbf{P}^{-1} = \mathbf{P}$.

Dacă se consideră un vector *arbitrar* **v**, definim

$\mathbf{w} = \mathbf{v} / \| \mathbf{v} \|_2$,

şi **P** devine:

$$\mathbf{P} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^T}{\| \mathbf{v} \|_2^2} \qquad (1)$$

Sau, notând

$$H = \tfrac{1}{2} \| \mathbf{v} \|_2^2 = \tfrac{1}{2} \mathbf{v}^T \mathbf{v} = \tfrac{1}{2} < \mathbf{v}, \mathbf{v} >, \qquad (2)$$

$$\mathbf{P} = \mathbf{I} - \frac{\mathbf{v}\mathbf{v}^T}{H}$$

Proprietate

Pentru un vector dat **x**, vrem să alegem **v** astfel ca să avem

$\mathbf{P}\mathbf{x} = \mu \mathbf{e}^{(1)}$,

unde $\mathbf{e}^{(1)}$ este prima coloană a matricii unitate $n \times n$. Adică:

$$\mathbf{P}\mathbf{x} = \begin{bmatrix} \mu \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Cu alte cuvinte, **P** roteşte **x** pe direcţia $\mathbf{e}^{(1)}$. Aceasta se realizează luând

$\mathbf{v} = \mathbf{x} \mp \| \mathbf{x} \|_2 \ \mathbf{e}^{(1)}$.

Rezultă

$\mathbf{P}\mathbf{x} = \mp \| \mathbf{x} \|_2 \ \mathbf{e}^{(1)}$

(Proprietatea se demonstrează astfel: se calculează $< \mathbf{v}-\mathbf{x}, \mathbf{v}-\mathbf{x} >$ şi se arată că avem $< \mathbf{v}, \mathbf{v} > = 2 < \mathbf{v}, \mathbf{x} >$; se calculează apoi, **Px**.)

■

1.2 Algoritm:

Algoritmul parcurge următoarele etape:

I. Reducerea lui **A** la forma tridiagonală dacă **A** este simetrică, sau la forma Hessenberg dacă **A** este nesimetrică.

II. Reducerea lui **A** la forma triunghiulară, sau bloc- triunghiulară (v. mai jos).

III. Descompunerea **QR**.

Algoritmul *QR* propriu-zis, constă în Etapele II şi III.

a) Etapa I se realizează înainte de algoritmul *QR* pentru a aduce matricea **A** la o forma care reduce efortul de calcul. Aceasta, pentru că *numărul de operaţii* pe un pas *k* al iteraţiei *QR* (Etapele II şi III), aplicată unei matrici $n \times n$ este de ordinul (Ralston & Rabinowitz, 1983):

- n^3 – pentru o matrice generală;
- n^2 – pentru o matrice Hessenberg;
- n – pentru o matrice tridiagonală.

Etapa I se realizează prin pre- şi post-multiplicare cu matrici Householder $\mathbf{P}_k$.

b) Etapa II se realizează prin premultiplicare cu matrici Householder $\mathbf{P}_k$, cum urmează:

$\mathbf{A}_1 = \mathbf{P}_1\mathbf{A}$; $\mathbf{A}_2 = \mathbf{P}_2\mathbf{A}_1 = \mathbf{P}_2\mathbf{P}_1\mathbf{A}$; …; $\mathbf{A}_{n-1} = \mathbf{P}_{n-1}\ldots\mathbf{P}_2\mathbf{P}_1\mathbf{A}$.

Matricea $\mathbf{A}_{n-1}$ are forma triunghiulară şi este matricea **R**.

Într-adevăr: Să notăm $\mathbf{Q} = \mathbf{P}_1\mathbf{P}_2\ldots\mathbf{P}_{n=1}$, avem $\mathbf{A}_{n-1} = \mathbf{Q}^T\mathbf{A}$. **Q** este o matrice ortogonală, deci avem $\mathbf{Q}\mathbf{A}_{n-1} = \mathbf{A}$. Cum **Q** este ortogonală şi $\mathbf{A}_{n-1}$ este superior triunghiulară, urmează că $\mathbf{R} = \mathbf{A}_{n-1}$ ■

Atunci, Etapele II şi III se realizează prin următorii paşi de iterare:

1) Pune $\mathbf{A}^{(1)} = \mathbf{A}$

- Triangularizează $\mathbf{A}^{(1)}$:

$$\mathbf{A}_1^{(1)} = \mathbf{P}_1^{(1)}\mathbf{A};\ \ \mathbf{A}_2^{(1)} = \mathbf{P}_2^{(1)}\mathbf{A}_1^{(1)};\ \ldots;\ \mathbf{A}_{n-1}^{(1)} = \mathbf{P}_{n-1}^{(1)}\mathbf{A}_{n-2}^{(1)}$$

- La sfârşitul pasului, avem:

$$\mathbf{R}^{(1)} = \mathbf{A}_{n-1}^{(1)};\quad \mathbf{Q}^{(1)} = \mathbf{P}_1^{(1)}\mathbf{P}_2^{(1)}\ldots\mathbf{P}_{n-1}^{(1)}.$$

2) Calculează $\mathbf{A}^{(2)} = \mathbf{R}^{(1)}\mathbf{Q}^{(1)}$

- Triangularizează $\mathbf{A}^{(2)}$:

$$\mathbf{A}_1^{(2)} = \mathbf{P}_1^{(2)}\mathbf{A}^{(2)};\ \ \mathbf{A}_2^{(2)} = \mathbf{P}_2^{(2)}\mathbf{A}_1^{(2)};\ \ldots;\ \mathbf{A}_{n-1}^{(2)} = \mathbf{P}_{n-1}^{(2)}\mathbf{A}_{n-2}^{(2)}$$

- Pune:

$$\mathbf{R}^{(2)} = \mathbf{A}_{n-1}^{(2)};\quad \mathbf{Q}^{(2)} = \mathbf{P}_1^{(2)}\mathbf{P}_2^{(2)}\ldots\mathbf{P}_{n-1}^{(2)}.$$

...

k) Calculează $\mathbf{A}^{(k)} = \mathbf{R}^{(k-1)}\mathbf{Q}^{(k-1)}$ $\qquad (k \geq 2)$

- Triangularizează $\mathbf{A}^{(k)}$:

$$\mathbf{A}_1^{(k)} = \mathbf{P}_1^{(k)}\mathbf{A}^{(k)};\ \ \mathbf{A}_2^{(k)} = \mathbf{P}_2^{(k)}\mathbf{A}_1^{(k)};\ \ldots;\ \mathbf{A}_{n-1}^{(k)} = \mathbf{P}_{n-1}^{(k)}\mathbf{A}_{n-2}^{(k)}$$

- Pune:

$$\mathbf{R}^{(k)} = \mathbf{A}_{n-1}^{(k)};\quad \mathbf{Q}^{(k)} = \mathbf{P}_1^{(k)}\mathbf{P}_2^{(k)}\ldots\mathbf{P}_{n-1}^{(k)}$$

...

■

Calculaţia continuă până când elementele triunghiului sub-diagonal în $\mathbf{A}_{n-1}^{(k)}$ sunt aproximativ zero. Atunci, valorile proprii sunt pe diagonala lui $\mathbf{A}_{n-1}^{(k)}$.

Mai precis, avem:

1) Dacă **A** are valori proprii *reale*, atunci:

A se reduce la forma triunghiulară. (calculaţia se termină când toate elementele sub-diagonale sunt aproximativ zero).

În particular, dacă **A** este simetrică, matricea se reduce la forma diagonală. (toate elementele non-diagonale vor fi aproximativ zero). Reamintim că o matrice simetrică are valori proprii reale.

2) Dacă **A** (nesimetrică) are *o pereche de valori proprii complexe* (conjugate), pe diagonală rămân submatrici blocuri 2×2, ale căror valori proprii sunt perechea de valori proprii conjugate ale lui **A**.

3) Dacă **A** (nesimetrică) are valori proprii λ_i şi $|\lambda_i|$ este de multiplicitate *m*, atunci $\mathbf{A}^{(k)}$ are un bloc diagonal de ordinul *m*, ale cărui valori proprii sunt λ_i.

(V. Wilkinson (1965), Ralston & Rabinowitz (1983), Meglicki (2001).)

Deci, cu excepţia cazului 3 cu $m > 2$ ($m > 2$ valori proprii *distincte şi de module egale*), pentru *k* suficient de mare, matricea $\mathbf{A}^{(k)}$ are următoarea formă:

$$\mathbf{A}^{(k)} = \begin{bmatrix} \lambda_1 & & & & & & \\ 0 & \lambda_2 & & & \mathbf{C} & & \\ \vdots & \vdots & \ddots & & & & \\ & & & \lambda_r & & & \\ & & & & \mathbf{B}_1 & & \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \\ 0 & 0 & \dots & 0 & & \dots & \mathbf{B}_s \end{bmatrix},$$

unde $\mathbf{B}_j = \begin{bmatrix} b_{11}^{(j)} & b_{12}^{(j)} \\ b_{21}^{(j)} & b_{22}^{(j)} \end{bmatrix}$, iar **C** conţine elementele superioare. Avem $r + 2s = n$.

Valorile λ_i şi blocurile $\mathbf{B}_j$, pot apare în orice ordine pe diagonală.

■

Un exemplu de matrice pentru cazul 3 cu $m > 2$, este (Wilkinson, 1965, [34], cu $\alpha = 1$):

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \alpha \\ \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 \end{bmatrix}$$

Valorile proprii sunt $\pm\alpha, \pm i\alpha$. Matricea constituie un bloc de ordinul 4, şi este invariantă la transformările QR.

Notă

Cazul 3 se poate evita în practică, prin algoritmul QR cu deplasare – v. mai jos

■

1.3 Algoritmul QR cu deplasare:

Presupunem că **A** are valori proprii *de module distincte* $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0$.

Se arată că viteza de convergenţă la zero a elementelor sub-diagonale, şi cea de convergenţă la valorile proprii a elementelor diagonale, depind de rapoartele $(\lambda_{i+1}/\lambda_i)^k,\ 1 \le i \le n-1.$

Dacă două valori proprii λ_i şi λ_{i+1} sunt apropiate una de alta, convergenţa va fi înceată. Atunci, se utilizează următoarea tehnică pentru a accelera convergenţa. Se aplică o deplasare s_k la valorile proprii (acestea devin $\lambda_i - s_k$), la fiecare etapă *k*.

Exemplu: $\lambda_i = 2.2, \lambda_{i+1} = 2.1$; avem $\lambda_{i+1}/\lambda_i = 0.95$. Cu $s = 2.0$, raportul are valoarea $0.1/0.2 = 0.5$.

Adică, punem

$\mathbf{A}_1 = \mathbf{A}$, şi

$$\mathbf{A}^{(k)} - s_k\mathbf{I} = \mathbf{Q}^{(k)}\mathbf{R}^{(k)},$$

$$\mathbf{A}^{(k+1)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)} + s_k\mathbf{I}$$

Există două strategii pentru a alege deplasarea s_k (s_k est o aproximaţie a lui λ_n):

1) $s_k = a_{nn}^{(k)}$

Pentru o *matrice simetrică* (reală), această strategie asigură o convergenţă cubică, chiar în prezenţa unor valori proprii multiple (Wilkinson, 1965).

2) Se calculează valorile proprii ale submatricii 2×2 cea mai de jos din $\mathbf{A}^{(k)}$:

$$\begin{bmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{bmatrix}$$

Dacă valorile proprii sunt reale, se alege:

s_k = valoarea proprie care este cea mai apropiată de $a_{nn}^{(k)}$.

Dacă valorile proprii sunt complexe, se ia:

s_k = partea reală a valorii proprii.

3) Experimentele numerice au condus la o a treia strategie, şi anume:

$$s_k = \min(a_{ii}^{(k)}), \quad i = \overline{1,n}$$

Pentru unele matrici simetrice, această strategie se dovedeşte cea mai bună, conducând la un număr mai mic de iteraţii, şi o eroare mai mică în $\mathbf{Ay} - \lambda\mathbf{y}$.

1.4 Programul QR

Programul QR din ANA, calculează sistemul propriu (valori şi vectori proprii), în succesiunea următoare:

I. Transformarea preliminară a matricii **A** (Opţional).

II. Valorile proprii, prin reducerea matricii la forma bloc-triunghiulară. (Programul recunoaşte numai blocuri 2×2.)

III. Vectorii proprii (Opţional).

Se calculează, mai întâi, vectorii proprii ai matricii bloc-triunghiulare. Aceştia se aduc, prin transformarea de similaritate, la vectorii proprii ai lui **A**.

IV. Rafinarea sistemului propriu prin iteraţie inversă (Opţional).

V. Verificarea sistemului propriu (Opţional):

Precizia sistemului propriu se verifică prin listarea erorii maxime (în modul), în $\mathbf{Ay} - \lambda\mathbf{y}$ (unde λ este valoarea proprie, iar **y** vectorul propriu asociat cu λ).

Exemplu – 1: Matrice simetrică

Considerăm matricea:

$$\mathbf{A} = \begin{bmatrix} 1 & -3 & -2 & 1 \\ -3 & 10 & -3 & 6 \\ -2 & -3 & 3 & -2 \\ 1 & 6 & -2 & 1 \end{bmatrix}$$

Programul QR, rulat cu *eps* = 1E-7 (toleranţă pentru elementele non-diagonale), *eps_lambda* = 1E-6 (toleranţa la λ), *deplasare* – strategia 1, produce următoarele rezultate (matricile sunt listate în format 1pG15.7, iar valorile proprii în format 1pG24.16):

```
Matrix A - Tridiagonal form:
  1.000000        3.741657      -4.0811097E-16  2.8277989E-16
  3.741657        2.785714      -5.246476       0.000000
 -4.0811097E-16  -5.246476       10.19927      -4.479586
  2.8277989E-16   0.000000      -4.479586       1.015014

 Iteration 19:
 Reduced Matrix - Block-Triangular Form
  14.32951        8.3206414E-08  -1.0862088E-10  6.0791490E-16

  8.3206414E-08   4.456957       -3.9733936E-03 -2.9646362E-11
 -1.0862122E-10  -3.9733936E-03  -3.415088      -2.2709916E-08

 -8.1045051E-19 -2.9646498E-11 -2.2709916E-08 -0.3713752

 Eigenvalues
 1   14.32950642539378
 2  -3.415090280621962
```

```
 3   4.456959098788067
 4  -0.3713752435599103
```

Programul calculează şi vectorii proprii **y**. Diferenţa maximă în **Ay** – λ**y** este 5.766E-08.

Programul rulat *cu deplasare* – strategia 3, termină în 9 iteraţii, cu diferenţa maximă 1.228E-12. Pentru acest exemplu, strategia 3 de alegere a deplasării este cea mai bună.

■

Exemplu – 1 bis: Rafinare

Considerăm matricea şi parametrii din Exemplu-1, şi facem rafinarea sistemului propriu cu *ns* = 4. Se obţin rezultatele (valorile proprii sunt listate în format 1pg24.16):

```
     Iterations      Test Value          Tolerance
 1   4               4.4408920985E-16    1.776E-15
 2   4               2.7194799110E-16    4.441E-16
 3   3                0.000000000        8.882E-16
 4   4               1.5324295440E-14(stationary)


Eigenvalues
 1   14.32950642539381
 2  -3.415090280621964
 3   4.456959098788065
 4 -0.3713752435599113
```

Diferenţa maximă în **Ay** – λ**y** este acum, 1.110E-15.

■

Exemplu – 2: Matrice nesimetrică

Considerăm matrice:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ 2 & 1 & 5 & 0 \\ 4 & 2 & 1 & 0 \end{bmatrix}$$

Programul QR, cu *eps* = 1E-7, *eps_lambda* = 1E-8, *deplasare* – strategia 1, produce rezultatele:

```
Matrix A - Hessenberg form:
   1.000000      -5.000000      -1.539516      -1.276672
  -6.000000       8.555556      0.1084754       3.698593
   0.000000      -5.918166      -2.168879      -1.142756
   0.000000       0.000000     -0.1427564       3.613324

Iteration 24:
 Reduced Matrix - Block-Triangular Form
   4.153893       10.18835       2.745848       2.753036
   5.464962       3.096039       3.428772       2.606012

  6.2957059E-08 -5.5017566E-09  0.1764519      0.1516238

 -2.9451447E-16  2.5737335E-17  1.5892009E-08   3.573617

   Eigenvalues
    1  -3.855588238007097
    2   11.10551971006653
    3  0.1764519119325265
    4   3.573616616008061
```

Diferenţa maximă în $\mathbf{Ay} - \lambda\mathbf{y}$ este 4.403E-08.

Cu strategia 3, programul termină în 22 iteraţii şi diferenţa maximă este 8.755E-08.

■

Exemplu – 2 bis: Rafinare

Considerăm matricea şi parametrii din Exemplu-2, şi facem rafinarea sistemului propriu cu *ns* = 4. Se obţin rezultatele:

```
     Iterations      Test Value          Tolerance

 1   4               3.1401849174E-16    4.441E-16

 2   4               2.2204460493E-16    1.776E-15

 3   5                0.000000000        2.776E-17

 4   5                0.000000000        4.441E-16


Eigenvalues

 1  -3.855588220333913

 2   11.10551973067810

 3  0.1764518729384592

 4   3.573616616717359
```

Diferenţa maximă în **Ay** – λ**y** este 1.776E-15.

■