Laborator 5

Calculul valorilor proprii

5.1 Valori şi vectori proprii

Definiţia 5.1 Fie $A \in \mathbf{C}^{n\times n}$. Un vector $x \in \mathbf{C}^{n\times n}$ este un *vector propriu* al matricei A, asociat *valorii proprii* $\lambda \in \mathbf{C}$, dacă sunt satisfăcute următoarele două proprietăţi

$$\begin{array}{ll} a) & x \neq 0 \\ b) & Ax = \lambda x. \end{array} \tag{5.1}$$

Câteva dintre cele mai importante proprietăţi ale valorilor proprii ale unei matrice pătrate sunt prezentate mai jos.

Teorema 5.2 *Fie $A \in \mathbf{R}^{n\times n}$. Numărul $\lambda \in \mathbf{C}$ este o valoare proprie a lui A dacă şi numai dacă matricea $\lambda I - A$ este singulară. Mai mult, matricea A sunt exact n valori proprii, incluzând multiplicităţile, care sunt zerourile* polinomului caracteristic

$$p(\lambda) = det(\lambda I_n - A). \tag{5.2}$$

Dacă $A \in \mathbf{R}^{n\times n}$, atunci valorile proprii complexe apar în perechi conjugate.

Notând cu

$$\lambda(A) = \{\lambda_1, \lambda_2, ..., \lambda_n\} = \{\lambda \in \mathbf{C} \mid det(\lambda I - A) = 0\} \tag{5.3}$$

mulţimea tuturor valorilor proprii ale matricei $A \in \mathbf{C}^{n\times n}$. Mulţimea$\lambda(A)$ se numeşte *spectrul de valori proprii al lui A*. Este, evident, util să definim transformările matriceale care conservă spectrul de valori proprii al unei matrice date.

Definiţia 5.3 Două matrice $A, B \in \mathbf{R}^{n\times n}$ se numesc *asemenea* dacă există o matrice nesingulară $T \in \mathbf{R}^{n\times n}$ astfel încât

$$B = TAT^{-1}. \tag{5.4}$$

Dacă T este o matrice ortogonală, atunci A şi B se numesc *ortogonal asemenea*.

Teorema 5.4 *Dacă $A, B \in \mathbf{R}^{n\times n}$ sunt asemenea atunci ele au acelaşi spectru de valori proprii*

$$\lambda(A) = \lambda(B). \tag{5.5}$$

Mai mult, dacă T este o transformare de asemănare ca în (5.4) şi dacă x_A este un vector propriu al lui A asociat valorii proprii $\lambda \in \lambda(A)$, atunci vectorul

$$x_B = Tx_A \tag{5.6}$$

este un vector propriu al matricei B, asociat aceleiaşi valori proprii.

5.2 Forma Schur (reală)

Forma Schur (reală) joacă un rol central în calculul eficient al valorilor proprii ale unei matrice.

Definiţia 5.5 O matrice $S \in \mathbf{R}^{n\times n}$ se numeşte *formă Schur reală (FSR)* dacă are structura

$$S = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1q} \\ 0 & S_{22} & \cdots & S_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \ldots & S_{qq} \end{bmatrix}, \tag{5.7}$$

unde $S_{ii} \in \mathbf{R}^{1\times 1}$ sau $S_{ii} \in \mathbf{R}^{2\times 2}$ şi toate blocurile diagonale 2×2 au valori proprii complexe.

O consecinţă imediată a structurii Schur este dată în propoziţia următoare.

Propoziţia 5.6 Fie $S \in \mathbf{R}^{n\times n}$ în FSR. Atunci:

$$\lambda(S) = \cup_{i=1}^{q} \lambda(S_{ii}). \tag{5.8}$$

Un rezultat important în calculul valorilor proprii este dat de următoarea teoremă.

Teorema 5.7 *Pentru orice matrice pătrată $A \in \mathbf{R}^{n\times n}$ există o matrice ortogonală $Q \in \mathbf{R}^{n\times n}$ astfel încât*

$$Q^T AQ = S \tag{5.9}$$

este in FSR.

Calculul (exact) al formei Schur (reale) pentru matrice de ordin superior lui 4 nu poate fi realizat într-un număr finit de operaţii elementare pentru că altfel s-ar contrazice un rezultat fundamental din algebră conform căruia rezolvarea unei ecuaţii algebrice de grad superior lui 4 nu este posibilă într-un număr finit de operaţii elementare.

Pe de altă parte, dacă se dispune de un vector propriu x, calculul valorii proprii asociate λ este imediat folosind formula

$$\lambda = \frac{x^T Ax}{x^T x}.$$

Mai mult, printr-o procedură de deflaţie, după determinarea unui vector propriu şi a valorii proprii asociate, problema îşi reduce dimensiunea. De aceea, există metode de calcul iterativ al unui vector propriu în absenţa cunoaşterii valorii proprii asociate, metode care stau la baza algoritmului QR, de calcul iterativ al formei Schur (reale). Prezentăm direct algoritmii corespunzători celor mai folosite astfel de metode: *metoda puterii* şi *metoda puterii inverse*.

5.2.1 Metoda puterii

Presupunem că matricea $A \in \mathbf{R}^{n\times n}$ are o valoare proprie dominantă, i.e. mai mare în modul decât toate celelalte, şi fie aceasta λ_1. Metoda puterii se bazează faptul că şirul vectorial

$$y^{(k)} = A^k y_0, \quad k = 1, 2, \cdots, \tag{5.10}$$

unde y_0 este un vector aleator ales, tinde, ca direcţie, către direcţia vectorului propriu asociat valorii proprii dominante. Introducând şi normarea vectorilor din acest şir, acesta poate fi definit recurent prin

$$y^{(k)} = \rho_k A y^{(k-1)}, k = 1, 2, \cdots \tag{5.11}$$

unde $\rho_k = \frac{1}{\|Ay^{(k-1)}\|}$ este un factor de scalare.

ALGORITM 5.8 *(Metoda puterii)* (Dată o matrice $A \in \mathbf{C}^{n\times n}$, un nivel de toleranţă $tol \in \mathbf{R}$, $tol < 1$, şi un număr maxim admis *maxiter* de iteraţii, algoritmul calculează un vector propriu unitar y asociat valorii proprii dominante a matricei date sau tipăreşte un mesaj dacă obiectivul nu a fost atins în numărul de iteraţii admis.)

1. Se alege aleator un vector $y \in \mathbf{R}^n$
2. $y \leftarrow y/\|y\|$
3. $i = 0, \quad e = 1$
4. `Cât timp` $e > tol$
    1. `Dacă` $i > maxiter$ `atunci`
        1. `Tipăreşte` 'S-a atins numărul maxim de iteraţii fără a se fi obţinut nivelul prescris al toleranţei.'
        2. `Stop`
    2. $z = Ay$
    3. $z \leftarrow z/\|z\|$
    4. $e = |1 - |z^T y|\,|$
    5. $y \leftarrow z$
    6. $i \leftarrow i + 1$.

Dacă matricea nu are o valoare proprie net dominantă, viteza de convergenţă poate fi nesatisfăcătoare, iar în cazul absenţei unei valori proprii dominante şirul poate fi divergent.

5.2.2 Metoda puterii inverse

Metoda puterii inverse pentru matricea A este, în fapt, metoda puterii pentru matricea $(\mu I - A)^{-1}$, unde μ este un scalar adecvat ales la fiecare pas, numit deplasare.

Prin urmare, metoda puterii inverse construieşte un şir de vectori cu relaţia de recurenţă

$$y^{(k)} = \rho_k(\lambda I - A)^{-1} y^{(k-1)}, \quad k = 1, 2, ... \tag{5.12}$$

Desigur, mijlocul eficient de implementare a iteraţiei (5.12), evită calculul matricei $(\lambda I - A)^{-1}$ ci apelează la rezolvarea unui sistem liniar, conform următoarei scheme.

1. Se rezolvă în raport cu $y^{(k)}$ sistemul liniar $(\lambda I - A)y^{(k)} = y^{(k-1)}$.
2. $y^{(k)} \leftarrow y^{(k)}/\|y^{(k)}\|$.

Metoda puterii inverse este una dintre cele mai bune metode iterative de calcul al unui vector propriu a unei matrice.

Cea mai bună alegere a deplasării este dată de aşa numitul *cât Rayleigh* definit de

$$\lambda = \frac{(y^{(k-1)})^T A y^{(k-1)}}{\|y^{(k-1)}\|^2} = (y^{(k-1)})^T A y^{(k-1)}. \tag{5.13}$$

Utilizând acelaşi criteriu de trunchiere ca şi la metoda puterii obţinem următorul algoritm.

ALGORITM 5.9 *(Metoda puterii inverse cu deplasare Rayleigh)*
(Dată o matrice $A \in \mathbf{R}^{n\times n}$, un nivel de toleranţă $tol \in \mathbf{R}$, $tol < 1$, şi un număr maxim admis *maxiter* de iteraţii, algoritmul calculează un vector propriu unitar y al matricei date sau tipăreşte un mesaj dacă obiectivul nu a fost atins în numărul admis de iteraţii.)

1. Se alege aleator un vector $y \in \mathbf{R}^n$.
2. $y \leftarrow y/\|y\|$
3. $i = 0, \quad e = 1$
4. `Cât timp` $e > tol$
    1. `Dacă` $i > maxiter$ `atunci`
        1. `Tipăreşte` 'S-a atins numărul maxim de iteraţii fără a se fi obţinut nivelul prescris al toleranţei.'
        2. `Stop`
    2. $\mu = y^T A y$
    3. Se rezolvă sistemul liniar $(\mu I_n - A)z = y$

4. $z \leftarrow z/\|z\|$
5. $e = |1 - |z^T y|\,|$
6. $y \leftarrow z$
7. $i \leftarrow i + 1$.

5.3 Algoritmul QR

Algoritmul **QR** este, în esenţă, o procedură de deflaţie iterativă care construieşte (recurent) un şir de matrice ortogonal asemenea cu matricea iniţială, şir care, în anumite condiţii, este convergent către forma Schur (reală).

În vederea minimizării efortului de calcul, într-o fază preliminară, matricea dată este adusă, prin transformări de asemănare ce implică un număr (teoretic) *finit* şi (practic) rezonabil de mic de operaţii, la cea mai apropiată structură posibilă de forma Schur (reală). Această structură este forma superior Hessenberg [1]. În continuare, structura Hessenberg este conservată de recurenţa fazei iterative a algoritmului. În acest fel, se obţine o importantă reducere a complexităţii unei iteraţii **QR**, fapt esenţial în economia algoritmului.

5.3.1 Reducerea la forma superior Hessenberg

Posibilitatea reducerii unei matrice $A \in \mathbf{R}^{n\times n}$ la forma superior Hessenberg, cu conservarea valorilor proprii, este dată de următorul rezultat.

Teorema 5.10 *Oricare ar fi matricea $A \in \mathbf{R}^{n\times n}$, există o matrice ortogonală $Q \in \mathbf{R}^{n\times n}$, calculabilă printr-o secvenţă finită de operaţii aritmetice, astfel încât matricea*

$$H = Q^T A Q \tag{5.14}$$

este superior Hessenberg.

Concret, reducerea la forma superior Hessenberg este bazată pe următoarea schemă de calcul

HQ 1. `Pentru` $k = 1 : n-2$
   1. Se calculează un reflector elementar U_{k+1} astfel încât $(U_{k+1}A)(k+2:n,k) = 0$.
   2. $A \leftarrow U_{k+1}A$
   3. $A \leftarrow AU_{k+1}$.

care suprascrie matricea A cu matricea Hessenberg rezultată

$$A \leftarrow H = U_{n-1}\cdots U_3 U_2 A U_2 U_3 \cdots U_{n-1} = Q^T A Q, \tag{5.15}$$

[1] Precizăm că matricea $H \in \mathbf{R}^{n\times n}$ este în formă superior Hessenberg dacă $h_{ij} = 0,\ \forall\, i > j+1$.

unde

$$Q = U_2 U_3 \cdots U_{n-1}. \tag{5.16}$$

Avându-se în vedere că reflectorul U_{k+1} are structura

$$U_{k+1} = \begin{bmatrix} I_k & 0 \\ 0 & \bar{U}_1 \end{bmatrix},$$

unde $\bar{U}_1$ este un reflector de ordin $n-k$ şi indice 1, rezultă imediat următoarele echivalenţe calculatorii:

$$A \leftarrow U_{k+1}A \qquad \equiv \qquad A(k+1:n,:) \leftarrow \bar{U}_1 A(k+1:n,:),$$

$$A \leftarrow AU_{k+1} \qquad \equiv \qquad A(:,k+1:n) \leftarrow A(:,k+1:n)\bar{U}_1.$$

Prin urmare, pentru obţinerea formei superior Hessenberg este suficient să ştim să "operăm" numai cu reflectori *de indice 1.* În acest scop introducem următoarele proceduri:

H (Procedură de calcul a elementelor definitorii u şi β ale unui reflector de indice 1 U_1 astfel încât $(U_1 x)(2:n) = 0$ şi de calcul $x \leftarrow U_1 x$, unde $x \in \mathbf{R}^n$ este un vector dat.)

1. $\sigma = \text{sign}(x_1)\sqrt{\sum_{i=1}^{n} x_i^2}$
2. $u_1 = x_1 + \sigma$
3. `Pentru` $i = 2:n$
   1. $u_i = x_i$
   2. $x_i = 0$
4. $\beta = u_1 \sigma$
5. $x_1 = -\sigma$

Sintaxa de apel a procedurii **H** va fi

$$[u, \beta, x] = \mathbf{H}(x).$$

Hs (Procedură de înmulţire la stânga a unei matrice $A \in \mathbf{R}^{n \times p}$ cu un reflector de indice 1, U_1, dat prin elementele sale definitorii u şi β, i.e. se calculează $A \leftarrow U_1 A$.)

1. `Pentru` $j = 1:p$
   1. $\tau = \left(\sum_{i=1}^{n} u_i a_{ij}\right)/\beta$
   2. `Pentru` $i = 1:n$
      1. $a_{ij} \leftarrow a_{ij} - \tau u_i$

Sintaxa de apel a procedurii **Hs** va fi

$$A = \mathbf{Hs}(u, \beta, A).$$

Hd (Procedură de înmulţire la dreapta a unei matrice $A \in \mathbf{R}^{m\times n}$ cu un reflector de indice 1, U_1, dat prin elementele sale definitorii u şi β, i.e. se calculează $A \leftarrow AU_1$.)

1. `Pentru` $i = 1 : m$
   1. $\tau = \left(\sum_{j=1}^{n} a_{ij}u_j\right)/\beta$
   2. `Pentru` $j = 1 : n$
      1. $a_{ij} \leftarrow a_{ij} - \tau u_j$

Sintaxa de apel a procedurii **Hd** va fi

$$A = \mathbf{Hd}(A, u, \beta).$$

Utilizând procedurile **H**, **Hs** şi **Hd**, se obţine imediat următorul algoritm.

ALGORITM 5.11 *(***HQ** – *Reducerea la forma superior Hessenberg)* (Dată o matrice $A \in \mathbf{R}^{n\times n}$, algoritmul calculează o secvenţă de reflectori U_2, U_3, $\cdots$, U_{n-1} astfel încât matricea transformată $A \leftarrow H = U_{n-1}^T \cdots U_3^T U_2^T A U_2 U_3 \cdots U_{n-1}$ este în forma superior Hessenberg. De asemenea se calculează matricea de transformare $Q = U_2U_3 \cdots U_{n-1}$.)

1. $Q = I_n$
2. `Pentru` $k = 1 : n-2$
   1. $[\, u, \beta, A(k+1:n, k)\,] = \mathbf{H}(A(k+1:n, k))$
   2. $A(k+1:n, k+1:n) = \mathbf{Hs}(u, \beta, A(k+1:n, k+1:n))$
   3. $A(1:n, k+1:n) = \mathbf{Hd}(A(1:n, k+1:n), u, \beta)$
   4. $Q(1:n, k+1:n) = \mathbf{Hd}(Q(1:n, k+1:n), u, \beta)$.

Pentru apelul algoritmului **HQ** va fi utilizată sintaxa generală

$$[\, Q, H\,] = \mathbf{HQ}(A),$$

care exprimă posibilitatea de a memora rezultatele în alte tablouri decât cele iniţiale deşi calculele se fac cu suprascrierea internă a matricei iniţiale.

5.3.2 Faza iterativă a algoritmului QR

Etapa iterativă a algoritmului **QR** utilizează, într-o manieră implicită, metodele puterii şi puterii inverse pentru reducerea unei matrice la forma Schur (reală).

Precizăm că din motive de simplitate vom prezenta exclusiv algoritmul QR cu deplasare explicită cu paşi simpli care "funcţionează" numai pentru matrice reale cu spectru real.

Presupunem că matricea $H \in \mathbf{R}^{n\times n}$ are o structură superior Hessenberg. Algoritmul **QR** cu deplasare explicită construieşte un şir de matrice

$$H = H_1, H_2, \cdots, H_k, H_{k+1}, \cdots \tag{5.17}$$

pe baza relaţiei de recurenţă

$$\left\{ \begin{array}{l} H_k - \mu_k I_n = Q_k R_k \\ H_{k+1} = R_k Q_k + \mu_k I_n \end{array} \right. , \quad k = 1, 2, \cdots, \quad H_1 = H, \tag{5.18}$$

unde scalarul μ_k, denumit *deplasare*, este folosit pentru asigurarea convergenţei. Valoarea optimă a deplasării este

$$\mu_k = H_k(n, n).$$

În prima relaţie (5.18) matricea $H_k - \mu_k I_n$ este factorizată QR. În relaţia a doua din (5.18) matricea succesor H_{k+1} se obţine înmulţind matricele Q_k şi R_k în ordine inversă şi anulând deplasarea prin adunarea matricei $\mu_k I_n$. Şirul (5.17), generat de (5.18), este denumit *şirul* **QR**. Corespunzător, tranziţia $H_k \to H_{k+1}$ se numeşte *un pas* sau *o transformare* **QR**.

Prezentăm mai jos un algoritm **QR** didactic [2] cu paşi simpli cu deplasare explicită în care se realizează şi o gestionare structurală prin monitorizarea elementelor subdiagonale care devin neglijabile. Controlul elementelor subdiagonale se face în ordine inversă (ţinând seama de vitezele diferite de anulare asimptotică). În momentul în care un element subdiagonal devine neglijabil el este anulat efectiv iar în continuare transformările **QR** se aplică submatricei principale cu dimensiunea redusă cu o unitate.

ALGORITM 5.12 *(**QR** – Calculul iterativ al formei Schur reale)* (Date o matrice $A \in \mathbf{R}^{n\times n}$ cu spectru real şi o toleranţă *tol* pentru anularea elementelor subdiagonale, algoritmul calculează matricea superior Hessenberg $H = Q^T A Q$, ortogonal asemenea cu A şi apoi suprascrie H cu matricele H_k din şirul **QR** cu paşi simpli cu deplasare explicită. Tipărirea lui H la fiecare iteraţie permite urmărirea convergenţei şirului **QR** către forma Schur reală.)

1. $[\,\tilde{Q}, H\,] = \mathbf{HQ}(A)$
2. $iter = 0$
3. $k = n$
4. `Cât timp` $k > 1$
   1. $\mu = H(k, k)$
   2. `Pentru` $i = 1 : k$
      1. $H(i, i) = H(i, i) - \mu$
   3. $[\,Q, R\,] = \mathbf{qr}(H(1 : k, 1 : k))$

[2] Algoritmul **QR** profesional foloseşte paşi dubli cu deplasare implicită şi toate procedurile apelate sunt optimizate atât din punctul de vedere al eficienţei cât şi cel al memoriei utilizate.

4. $H(1:k, 1:k) = R * Q$
5. `Pentru` $i = 1 : k$
    1. $H(i, i) = H(i, i) + \mu$
6. `Dacă` $k < n$
    1. $H(1:k, k+1:n) = Q^T H(1:k, k+1:n)$
7. $\tilde{Q}(:, 1:k) = \tilde{Q}(:, 1:k)Q$
8. $iter = iter + 1$
9. `Tipăreşte` $iter$ `şi` H
10. `Dacă` $|H(k, k-1)| < tol$
    1. $H(k, k-1) = 0$
    2. $k = k - 1$

Observaţii.

1. Dacă nu aveţi timp să scrieţi o procedură eficientă de factorizare QR a unei matrice superior Hessenberg puteţi folosi funcţia `qr` din MATLAB pentru factorizarea QR a matricelor generale.

2. Dacă nu v-a reuşit obţinerea unui program de reducere la forma superior Hessenberg, în locul funcţiei dv. **HQ** puteţi utiliza funcţia `hess` din MATLAB.

3. La instrucţiunea 4.10, pentru neglijarea elementelor subdiagonale se preferă satisfacerea unei toleranţe relative, e.g. a unei condiţii de forma

$$|H(k, k-1)| < tol\,(|H(k-1, k-1)| + |H(k, k)|)\,.$$

5.4 Sarcini de lucru

5.4.1 A. In laborator

1. Se vor edita şi testa programele MATLAB pentru implementarea metodelor puterii şi puterii inverse.

2. Se va edita şi testa un program MATLAB pentru implementarea algoritmului HQ de reducere la forma superior Hessenberg a unei matrice prin transformări ortogonale de asemănare. Se va compara rezultatul obţinut cu cel oferit de funcţia MATLAB `hess`.

3. Se va edita şi testa un program MATLAB pentru implementarea algoritmului QR de reducere a unei matrice cu spectru real în forma superior Hessenberg la forma Schur (reală). Programul va afişa matricea curentă din şirul QR pentru a se putea urmări anularea asimptotică a elementelor subdiagonale.

   Pentru aceeaşi matrice se vor utiliza mai multe niveluri de toleranţă *tol* (e.g. $1.e-15$, $1.e-10$, $1.e-5$) şi se vor compara numerele de iteraţii necesare pentru aducerea la forma Schur.

   Observaţie. Pentru crearea unei matrice dense cu spectru impus (în particular real) se poate proceda în felul următor: dacă $L = \{ l_1, l_2, \cdots, l_n \}$ este

spectrul (real) impus atunci următorul program MATLAB crează o matrice A densă cu spectrul L:

```
L=[l_1, l_2, ..., l_n];
A=diag(L);
T=rand(n);
A=T*A/T;
```

Se recomandă impunerea unor valori proprii uşor de recunoscut (de exemplu, numere întregi).

5.4.2 B. Acasă

1 Se va edita şi testa un program MATLAB eficient de triangularizare ortogonală cu rotaţii a unei matrice superior Hessenberg şi un program MATLAB pentru implementarea algoritmului **QR** cu deplasare explicită cu paşi simpli, de reducere a unei matrice cu spectru real în forma superior Hessenberg la forma Schur (reală) fără a calcula explicit matricea Q din factorizarea QR a matricei $H - \mu I$.

2 Se va edita şi testa un program MATLAB pentru implementarea algoritmului QR cu deplasare *implicită* cu paşi simpli, de reducere a unei matrice cu spectru real în forma superior Hessenberg la forma Schur (reală). Programul va afişa matricea curentă din şirul **QR** pentru a se putea urmări anularea asimptotică a elementelor subdiagonale.