# Seminar 4

# Calculul valorilor şi vectorilor proprii

Valorile proprii ale unei matrice joacă un rol important în multe domenii cum ar fi teoria stabilităţii. De asemenea, ele sunt legate îndeaproape de problema rezolvării ecuaţiilor algebrice. Prin urmare, un calcul corect al valorilor şi vectorilor proprii este de mare importanţă. Din cauza faptului că valorile proprii sunt rădăcinile ecuaţiei polinomiale caracteristice, calculul lor pentru o matrice de ordin mai mare decât patru nu poate fi făcut exact printr-o secvenţă finită de operaţii aritmetice şi este nevoie de un proces infinit care trebuie să fie trunchiat astfel încât să se obţină o aproximare acceptabilă. Cel mai bun proces de acest tip este algoritmul QR care calculează o formă Schur aproximativă a matricei date. Acest seminar este dedicat diferitelor aspecte numerice ale calculului valorilor proprii şi ale implementărilor algoritmului QR.

## 4.1 Preliminarii

Fie $A \in \mathbb{R}^{n\times n}$. Un vector $x \in \mathbb{C}^n$ este un *vector propriu* al lui $A$ şi $\lambda \in \mathbb{C}$ este *valoarea proprie* asociată dacă sunt satisfăcute următoarele condiţii

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

Astfel, numărul $\lambda \in \mathbf{C}$ este o valoare proprie a lui $A$ dacă şi numai dacă matricea $\lambda I - A$ este singulară, i.e. valorile proprii sunt zerourile *polinomului caracteristic*

$$p(\lambda) = \det(\lambda I_n - A),$$

şi matricea $A$ are exact $n$ valori proprii, incluzând multiplicităţile. În mod evident, valorile proprii complexe ale unei matrice reale apar sub forma unor perechi conjugate. Mulţimea valorilor proprii ale unei matrice $A$ se notează cu $\lambda(A)$ şi se numeşte *spectrul (valorilor proprii)* ale lui $A$.

Dacă $x$ este un vector propriu al lui $A$, vectorul $y = \alpha x$ este de asemenea un vector propriu al lui $A$, asociat aceleiaşi valori proprii pentru orice scalar $\alpha$ diferit de zero. Deci vectorii proprii sunt determinaţi doar prin direcţia lor.

Valorile proprii sunt conservate de aşa numitele *transformări de asemănare* definite prin

$$B = TAT^{-1},$$

unde $T$ este o matrice nesingulară. Dacă $T$ este o matrice ortogonală atunci $A$ şi $B$ se numesc *ortogonal asemenea*. Dacă $A, B$ sunt asemenea atunci ele au acelaşi spectru de valori proprii $\lambda(A) = \lambda(B)$ iar vectorii proprii corespunzători sunt legaţi prin relaţia $x_B = Tx_A$.

Cea mai simplă formă care poate fi obţinută printr-o transformare de asemănare este aşa numita *formă canonică Jordan*. Dacă o matrice $n \times n$ are $n$ vectori proprii liniari independenţi, atunci forma sa canonică Jordan este diagonală şi matricea se numeşte simplă sau diagonalizabilă. Din păcate, forma canonică Jordan este foarte sensibilă la perturbări numerice. De aceea, în calculul numeric este preferată o formă mai robustă şi anume forma Schur (reală) (FSR) care poate fi obţinută printr-o transformarea *ortogonală* de asemănare:

$$S = Q^T AQ = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1q} \\ 0 & S_{22} & \cdots & S_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \ldots & S_{qq} \end{bmatrix},$$

unde $S_{ii} \in \mathbf{R}^{1\times 1}$ or $S_{ii} \in \mathbf{R}^{2\times 2}$ şi toate blocurile diagonale $2 \times 2$ au valori proprii complexe. Valorile reale ale unei matrice FSR pot fi determinate printr-o simplă inspecţie a diagonalei principale iar cele complexe pot fi calculate prin rezolvarea ecuaţiilor algebrice de gradul doi corespunzătoare $\det(\lambda I - S_{ii}) = 0$.

Fie o pereche (valoare proprie, vector propriu)=$(\lambda, x)$, (un sistem propriu) al unei matrice $A$. Dacă unul dintre cele două elemente este cunoscut celălalt poate fi calculat cu uşurinţă (vezi problema rezolvată 4.1). Dacă nici valoarea proprie nici vectorul propriu nu sunt cunoscute, problema calculului lor devine una foarte dificilă. Motivul stă în imposibilitatea rezolvării ecuaţiilor algebrice de grad mai mare decât patru prin secvenţe finite de operaţii elementare.

Cele mai folosite metode pentru calculul iterativ al vectorilor proprii (fără a se cunoaşte valoarea proprie asociată) sunt *metoda puterii* şi *metoda puterii inverse*. Aceste metode sunt implementate în mod profesional într-o faimoasă procedură iterativă numită *algoritmul QR*. Algoritmul QR construieşte iterativ o secvenţă de matrice ortogonal asemenea, convergentă către o formă Schur a matricei originale. Proprietăţile excelente de convergenţă se datorează folosirii unor deplasări optimale şi unei serii de subtilităţi tehnice cum ar fi utilizarea exclusivă a formei Hessenberg şi monitorizarea structurală. Toate variantele algoritmului QR sunt organizate în două etape:

1. Reducerea la forma superior Hessenberg printr-o transformare ortogonală de asemănare $H = U^T AU$ calculată într-un număr finit de flopi ($O(\frac{5}{3}n^3)$) (algoritmul HQ).

2. Reducerea iterativă la forma Schur, i.e. construirea iterativă a unui şir de matrice $H_1 = H, H_2, \ldots, H_k, \ldots$, convergent către o formă Schur a lui $A$. Pentru o matrice cu un spectru real, fiecare iteraţie QR este definită prin:

   1. $[\, Q_k, R_k \,] = \mathbf{qr}(H_k - \mu_k I_n)$ % Factorizarea QR a lui $H_k - \mu_k I_n$
   2. $H_{k+1} \leftarrow R_k * Q_k + \mu_k I_n$,

   unde $\mu_k = H_k(n, n)$ este deplasarea Rayleigh. Dacă matricea $A$ are valori proprii complexe, atunci se foloseşte o variantă mai complicată cu paşi dubli şi deplasări implicite (vezi cursul).

## 4.2 Probleme rezolvate

### 4.2.1 Aspecte teoretice

**Problema 4.1** Fie $A, B \in \mathbb{R}^{n\times n}$ două matrice asemenea, i.e. există o matrice nesingulară $T \in \mathbb{R}^{n\times n}$ astfel încât $B = T^{-1}AT$. Demonstraţi că

$$\textbf{a}. \quad \det A = \det B, \qquad \textbf{b}. \quad \mathrm{tr}A = \mathrm{tr}B,$$

unde $\mathrm{tr}A = \sum_{i=1}^{n} a_{ii}$ este, prin definiţie, urma lui $A$. Ca aplicaţie, demonstraţi că dacă $\lambda(A) = \{\lambda_i, \lambda_i, \ldots, \lambda_i\}$ este spectrul valorilor proprii ale lui $A$, atunci

$$\textbf{c}. \quad \det A = \prod_{i=1}^{n} \lambda_i, \qquad \textbf{d}. \quad \mathrm{tr}A = \sum_{i=1}^{n} \lambda_i.$$

**Rezolvare.** Este bine-cunoscut că $\det(GH) = \det G \det H$ pentru toate matricele pătrate $G$ şi $H$. Evident, acest lucru implică faptul că pentru toate matricele nesingulare $G$ avem $\det G^{-1} = \dfrac{1}{\det G}$. **a.** Din $B = T^{-1}AT$ rezultă că $\det B = \dfrac{1}{\det T} \det A \det T = \det A$.

**b.** Să notăm $X = T^{-1}$. Atunci $XT = TX = I_n$, i.e. $X(i,:)T(:,j) = T(i,:)X(:,j) = \delta_{ij}$, unde scalarul $\delta_{ij} = 0$ pentru toţi $i \neq j$ şi $\delta_{ij} = 1$ dacă $i = j$. Prin urmare

$$\mathrm{tr}B = \sum_{k=1}^{n} B(k,k) = \sum_{k=1}^{n} X(k,:)AT(:,k) = \sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} X(k,i)A(i,j)T(j,k) =$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} A(i,j)\sum_{k=1}^{n} T(j,k)X(k,i) = \sum_{i=1}^{n}\sum_{i=1}^{n} A(i,j)T(j,:)X(:,i) =$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} A(i,j)\delta_{ij} = \sum_{i=1}^{n} A(i,i) = \mathrm{tr}A.$$

**c.** Fie $S = Q^T AQ$ forma Schur reală pentru $A$. Atunci $\det A = \det S = \prod_{i=1}^{p} \det S_{ii}$. Dacă $S_{ii}$ este un bloc scalar $\det S_{ii} = S_{ii}$; dacă $S_{ii}$ este un bloc $2 \times 2$, atunci $\det S_{ii} = S_{ii}(1,1)S_{ii}(2,2) - S_{ii}(1,2)S_{ii}(2,1) = \lambda_1(S_{ii})\lambda_2(S_{ii})$. Astfel, $\det A = \prod_{i=1}^{p} \Pi_{j=1}^{n_i} \lambda_j(S_{ii}) = \prod_{i=1}^{n} \lambda_i(A)$.

**d.** Fie din nou $S = Q^T AQ$ forma Schur reală pentru $A$. Atunci $\mathrm{tr}A = \mathrm{tr}S = \sum_{i=1}^{p} \mathrm{tr}S_{ii}$. Dacă $S_{ii}$ este un bloc scalar $\mathrm{tr}S_{ii} = S_{ii}$; dacă $S_{ii}$ este un bloc $2\times 2$, atunci $\mathrm{tr}S_{ii} = S_{ii}(1,1) + S_{ii}(2,2) = \lambda_1(S_{ii}) + \lambda_2(S_{ii})$. Astfel, $\mathrm{tr}A = \sum_{i=1}^{p}\sum_{j=1}^{n_i} \lambda_j(S_{ii}) = \sum_{i=1}^{n} \lambda_i(A)$.

**Problema 4.2** Fie $A \in \mathbb{R}^{n\times n}$ şi $\lambda(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ spectrul valorilor sale proprii. **a.** Demonstraţi că valorile proprii sunt situate în următorul domeniu al planului complex

$$\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i,$$

unde $\mathcal{D}_i$ sunt discurile

$$\mathcal{D}_i = \{z \in \mathbb{C} \mid |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|\}, \quad i = 1 : n,$$

numite *discurile lui Gershgorin*.

**b.** Numărul $\rho(A) \stackrel{\text{def}}{=} \max_{i=1:n}(|\lambda_i|)$ se numeşte *raza spectrală* a lui $A$. Demonstraţi inegalitatea

$$\rho(A) \leq \|A\|,$$

unde $\|\cdot\|$ este o familie oarecare de norme matriceale consistente.

**c.** Demonstraţi următoarele inegalităţi

$$\mathbf{c_1}. \quad |\det A| \leq \prod_{i=1}^{n} \left( \sum_{j=1}^{n} |a_{ij}| \right), \qquad \mathbf{c_2}. \quad |\det A| \leq \prod_{j=1}^{n} \left( \sum_{i=1}^{n} |a_{ij}| \right).$$

**Rezolvare. a.** Fie $x$ un vector propriu asociat valorii proprii $\lambda \in \lambda(A)$. Atunci linia $i$ a egalităţii vectoriale $Ax = \lambda x$ poate fi scrisă sub forma

$$(\lambda - a_{ii})x_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}x_j.$$

Prin urmare, este evident că $|\lambda - a_{ii}||x_i| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}||x_j|$. Alegând acum linia $i$ astfel încât $|x_i| = \max_{k=1:n}(|x_k|) \neq 0$, i.e. $\dfrac{|x_j|}{|x_i|} \leq 1$ pentru toţi $j$, avem

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \frac{|x_j|}{|x_i|} \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|,$$

i.e. $\lambda \in \mathcal{D}_i$.

**b.** Datorită consistenţei familiei de norme matriceale, pentru toate $\lambda \in \lambda(A)$ şi toţi vectorii proprii asociaţi $x$ cu $\|x\| = 1$, avem $|\lambda| = \|\lambda x\| = \|Ax\| \leq \|A\|\|x\| = \|A\|$. Prin urmare inegalitatea este adevărată.

**c.** Să demonstrăm mai întâi inegalitatea $\mathbf{c_1}$. Dacă $A$ are (cel puţin) o linie zero, atunci inegalitatea devine egalitate. Dacă nu este cazul, să definim $\delta_i = \sum_{j=1}^{n} |a_{ij}| > 0$, $i = 1 : n$, şi $D = \text{diag}(\delta_1, \delta_2, \ldots, \delta_n)$. Matricea $B = D^{-1}A$ are $\rho(B) \leq \|B\|_\infty \leq 1$. Prin urmare, $|\det B| = \prod_{i=1}^{n} |\lambda_i(B)| \leq 1$. Inegalitatea dorită se obţine din $|\det A| = |\det D| \cdot |\det B| \leq |\det D|$. Pentru a demonstra $\mathbf{c_2}$ aplicăm inegalitatea din cazul $\mathbf{c_1}$ matricei $A^T$.

**Problema 4.3** Fie $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times m}$ cu $m > n$. Demonstraţi că $\lambda(AB) = \lambda(BA) \cup \{0, 0, \ldots, 0\}$, i.e. că, în particular, matricele $AB$ şi $BA$ au aceleaşi valori proprii nenule.

**Rezolvare.** Matricele $(m+n) \times (m+n)$ $C = \begin{bmatrix} AB & 0 \\ B & 0 \end{bmatrix}$ şi $D = \begin{bmatrix} 0 & 0 \\ B & BA \end{bmatrix}$ sunt asemenea deoarece $D = TCT^{-1}$ cu $T = \begin{bmatrix} I_m & A \\ 0 & I_n \end{bmatrix}$ având ca inversă $T^{-1} = \begin{bmatrix} I_m & -A \\ 0 & I_n \end{bmatrix}$

(verificaţi!). Prin urmare $\lambda(C) = \lambda(D)$. Dar $C$ şi $D$ sunt matrice bloc inferior triunghiulare. Astfel, $\lambda(C) = \lambda(AB) \cup \{0, 0, \ldots, 0\}$ şi $\lambda(D) = \{0, 0, \ldots, 0\} \cup \lambda(BA)$. Atunci, deoarece $m > n$, $\lambda(AB) = \lambda(BA) \cup \{0, 0, \ldots, 0\}$. Evident, dacă $m = n$ atunci $\lambda(AB) = \lambda(BA)$.

**Problema 4.4** **a.** Fie $\lambda_1$, $\lambda_2$ două valori proprii diferite ale matricei $A \in \mathbb{R}^{n\times n}$ şi $x_1$ un vector propriu al lui $A$ asociat lui $\lambda_1$ şi $y_2$ un vector propriu al lui $A^T$ asociat lui $\lambda_2$. Demonstraţi că cei doi vectori sunt ortogonali i.e. $y_2^T x_1 = 0$.

**b.** Dacă $\lambda \in \lambda(A)$ este o valoare proprie distinctă a lui $A$ şi $x$, respectiv $y$, sunt vectori proprii la dreapta, respectiv la stânga, asociaţi lui $\lambda$, atunci $y^T x \neq 0$. Daţi un exemplu în care această proprietate nu este satisfăcută dacă $\lambda$ nu este o valoare proprie distinctă.

**Rezolvare.** **a.** Avem $Ax_1 = \lambda_1 x_1$ şi $A^T y_2 = \lambda_2 y_2$ sau $y_2^T A = \lambda_2 y_2^T$. Prin urmare $\lambda_1 y_2^T x_1 = \lambda_2 y_2^T x_1$ sau $(\lambda_1 - \lambda_2) y_2^T x_1 = 0$ care implică $y_2^T x_1 = 0$ din cauză că $(\lambda_1 - \lambda_2) \neq 0$.

**b.** Fără să pierdem caracterul general, putem presupune că $\|x\|_2 = 1$. Conform cu lema de deflaţie ortogonală, dacă matricea $[\, x \ \tilde{X} \,]$ este ortogonală, atunci $B = X^T A X = \begin{bmatrix} \lambda & b^T \\ 0 & C \end{bmatrix}$. Acum, dacă vectorul $y$ este un vector propriu la stânga al lui $A$ (i.e. $y^T A = \lambda y^T$, sau $y^T A = \lambda y^T$; astfel, un vector propriu la stânga al lui $A$ este un vector propriu al lui $A^T$), atunci $y^T X B = y^T X X^T A X = y^T A X = \lambda y^T X$, deci $z^T B = \lambda z^T$ unde $z = X^T y$. Prin urmare $z$ este un vector propriu la stânga al lui $B$. Dar $\lambda$ este o valoare proprie distinctă, deci matricea $\lambda I_{n-1} - C$ este nesingulară. Rezultă că $z(2:n) = (\bar{\lambda} I_{n-1} - C^T)^{-1} b z_1$, cu $z_1 = x^T y$. Dacă $z_1 = 0$, atunci $z(2:n) = 0$ şi, prin urmare, $z = 0$. Dar un vector propriu nu poate să fie zero. Astfel $z_1 = x^T y \neq 0$. Drept exemplu pentru situaţia când valoarea proprie nu este distinctă şi condiţia nu este satisfăcută, fie matricea $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ ai cărei vectori proprii (la dreapta) au forma $x = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$ şi cei la stânga au forma $y = \begin{bmatrix} 0 \\ \beta \end{bmatrix}$, $\alpha, \beta \in \mathbb{R}$, $\alpha \neq 0$, $\beta \neq 0$; astfel $y^T x = 0$.

### 4.2.2 Aspecte calculatorii

**Problema 4.5** Fie $A \in \mathbb{R}^{m\times n}$ o matrice dată.

**a.** Presupunând că vectorul propriu $x \in \mathbb{R}^n$ al lui $A$ este cunoscut, scrieţi un algoritm pentru calculul valorii proprii asociate $\lambda$.

**b.** Presupunând că valoarea proprie $\lambda \in \lambda(A)$ este cunoscută, scrieţi un algoritm pentru calculul unui vector propriu asociat.

**Rezolvare.** **a.** Avem $\lambda x = Ax$, deci $\lambda x^T x = x^T A x$ şi, din cauză că $x \neq 0$, $x^T x = \|x\|^2 \neq 0$. Prin urmare

$$\lambda = \frac{x^T A x}{x^T x} = \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j}{\sum_{i=1}^n x_i^2}$$

şi algoritmul este

1. $\sigma = 0$
2. $\tau = 0$

3. **pentru** $i = 1 : n$
   1. $y_i = 0$
   2. **pentru** $j = 1 : n$
      1. $y_i = y_i + a_{ij}x_j$
   3. $\sigma = \sigma + x_i y_i$
   4. $\tau = \tau + x_i^2$
2. $\lambda = \sigma/\tau$

**b.** Dacă o valoare proprie $\lambda \in \lambda(A)$ este cunoscută, atunci un vector propriu asociat poate fi calculat prin calculul unei soluţii nenule a sistemului liniar omogen $(\lambda I - A)x = 0$. Pentru aceasta vom aplica eliminarea gaussiană cu pivotare parţială (algoritmul GPP) matricei $\lambda I - A$

$$M_{n-1}P_{n-1} \cdots M_2 P_2 M_1 P_1 (\lambda I - A) = U$$

unde $U$ este superior triunghiulară. Evident, soluţiile sistemului $Ux = 0$ sunt soluţii ale sistemului $(\lambda I - A)x = 0$. Matricea superior triunghiulară $U$ este singulară deoarece matricea $\lambda I - A$ este singulară. Astfel, matricea $U$ are cel puţin un element diagonal nul. Fie $u_{kk} = 0$ primul element diagonal nul a lui $U$. Sistemul $Ux = 0$ poate fi scris sub forma

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & 0 & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix} \begin{bmatrix} x' \\ x_k \\ x''' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

unde $U_{11} = U(1 : k-1, 1 : k-1)$ este un bloc superior triunghiular nesingular, $U_{12} = U(1 : k-1, k)$, $U_{23} = U(k, k+1 : n)$, $U_{33} = U(k+1 : n, k+1 : n)$. O soluţie nenulă poate fi obţinută alegând

$$x''' = 0, \qquad x_k = \alpha \neq 0, \qquad x' = -U_{11}^{-1}U_{12}\alpha$$

unde $\alpha$ este un scalar arbitrar nenul.

Prin urmare, un algoritm pentru calculul unui vector propriu, atunci când valoarea proprie asociată este cunoscută, este

1. $[\bar{M}, p, U]$ =GPP$(\lambda I - A)$
2. $k = 1$
3. **cât timp** $u_{kk} \neq 0$
   1. $k \leftarrow k + 1$
4. **pentru** $i = k + 1 : n$
   1. $x_i = 0$
5. $x_k = 1$
6. **dacă** $k > 1$
   1. $x(1 : k-1)$ =UTRIS$(U(1 : k-1, 1 : k-1), -U(1 : k-1, k))$

În algoritmul de mai sus scalarul $\alpha$ a fost considerat egal cu 1.

**Problema 4.6 a.** Calculaţi forma Schur reală a matricei $A = \begin{bmatrix} 1 & -2 \\ 2 & -3 \end{bmatrix}$.

**b.** Calculaţi forma Schur reală a matricei $A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$.

**c.** Presupunând că o matrice dată $A$ are un spectru real (i.e. $\lambda(A) \subset \mathbb{R}$) şi că dispunem de o procedură cu sintaxa

$$x = \textbf{eigenvector}(A)$$

pentru calculul unui vector propriu al matricei $A$ (fără a se cunoaşte valoarea proprie asociată), scrieţi un algoritm pentru calculul unei forme Schur a lui $A$.

**Rezolvare. a.** Polinomul caracteristic al lui $A$ este $p(\lambda) = \det(\lambda I_2 - A) = (\lambda - 1)(\lambda + 3) + 4 = (\lambda - 1)^2$, deci valorile proprii ale lui $A$ sunt ambele egale cu $-1$. Sistemul $Ax = -x$ conduce la $x_1 = x_2$, astfel că toţi vectorii proprii ai lui $A$ sunt de forma $x = \alpha \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\alpha \neq 0$. Pentru a aplica lema de deflaţie ortogonală, fie vectorul propriu de normă euclidiană unitate $x = (1/\sqrt{2}) \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Considerând matricea ortogonală $Q = [x\ y]$, unde e.g. $y = (1/\sqrt{2}) \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ (având, evident, $\|y\| = 1$), rezultă $S = Q^T A Q = \begin{bmatrix} -1 & 4 \\ 0 & -1 \end{bmatrix}$, care este o formă Schur a lui $A$. Observaţi că forma Schur poate fi obţinută pe cale indicată chiar dacă $A$ este defectivă (i.e. nu este simplă).

**b.** Polinomul caracteristic al lui $A$ este $p(\lambda) = \det(\lambda I_2 - A) = (\lambda - 1)^2 + 1$, deci valorile proprii ale lui $A$ nu sunt reale. Astfel o formă Schur reală (FSR) a lui $A$ este chiar $A$.

**c.** Vom aplica în mod sistematic lema de deflaţie ortogonală. După cum se demonstrează la curs, dacă $x$ este un vector propriu de normă euclidiană unitate al lui $A$ şi $Q = [\,x\ \ Y\,]$ este o matrice ortogonală atunci matricea

$$B = Q^T A Q = \begin{bmatrix} \lambda & b^T \\ 0 & C \end{bmatrix}$$

este în formă Schur în prima coloană. Pentru a continua, putem aplica aceeaşi procedură blocului $C$, etc. Algoritmul dorit se poate baza pe următoarea schemă de calcul:

1. **pentru** $k = 1 : n-1$
   1. $\tilde{x}_k = \textbf{eigenvector}(A(k:n, k:n))$
   2. $\tilde{x}_k = \dfrac{\tilde{x}_k}{\|\tilde{x}_k\|}$
   3. Se calculează o matrice ortogonală $\tilde{Q}_k$ astfel încât $\tilde{Q}_k e_1 = \tilde{x}_k$
   4. $A(k:n, k:n) = \tilde{Q}_k^T A(k:n, k:n)$
   5. $A(:, k:n) = A(:, k:n)\tilde{Q}_k$.

În final, matricea $A$ va fi suprascrisă cu forma sa Schur.

Pentru a detalia schema de mai sus, remarcaţi faptul că matricea $\tilde{Q}_k$ din instrucţiunea 1.3 poate fi un reflector $U = I - uu^T/\beta$ astfel încât $(U\tilde{x}_k)(2 : n-k+1) = 0$. Folosind procedurile cunoscute pentru calculul vectorului $u$ şi al scalarului $\beta$ ce defineşte reflectorul şi pentru înmulţirea la stânga şi la dreapta a unei matrice cu un reflector şi folosind un vector generic $x$ pentru $\tilde{x}_k$, algoritmul detaliat devine:

1. **pentru** $k = 1 : n-1$
   1. $x = \textbf{eigenvector}(A(k:n, k:n))$
   2. $\alpha = \sqrt{\sum_{i=1}^{n-k+1} x_i^2}$

3. **pentru** $i = 1 : n-k+1$
   1. $x_i \leftarrow x_i/\alpha$
4. $u_1 = x_1 - 1, \qquad \%\ \sigma = -1$
5. **pentru** $i = 2 : n-k+1$
   1. $u_i = x_i$
6. $\beta = -u_1, \qquad \%\ \sigma = -1$
7. **pentru** $j = k : n$
   1. $\tau = \left(\sum_{i=k}^{n} u_{i-k+1}a_{ij}\right)/\beta$
   2. **pentru** $i = k : n$
      1. $a_{ij} \leftarrow a_{ij} - \tau u_{i-k+1}$
8. **pentru** $i = 1 : n$
   1. $\tau = \left(\sum_{j=k}^{n} a_{ij}u_{j-k+1}\right)/\beta$
   2. **pentru** $j = k : n$
      1. $a_{ij} \leftarrow a_{ij} - \tau u_{j-k+1}$.

**Problema 4.7 a.** Fie $A = \begin{bmatrix} \alpha & \gamma \\ \gamma & \beta \end{bmatrix} \in \mathbf{R}^{2\times 2}$ o matrice simetrică. Daţi un algoritmu pentru calculul valorilor şi vectorilor proprii ai lui $A$. Reţineţi faptul că valorile proprii sunt reale şi vectorii proprii sunt ortogonali.

**b.** Arătaţi că o matrice simetrică reală are spectrul real şi vectori proprii ortogonali doi câte doi. Ce simplificări ale calculului apar în algoritmul QR simetric faţă de cazul nesimetric?

**Rezolvare. a.** Polinomul caracteristic al lui $A$ este $\det(\lambda I_2 - A) = (\lambda-\alpha)(\lambda-\beta)-\gamma^2 = \lambda^2 - (\alpha+\beta)\lambda + \alpha\beta - \gamma^2$ şi discriminantul său este $\delta = (\alpha+\beta)^2 - 4(\alpha\beta-\gamma^2) = (\alpha-\beta)^2 + 4\gamma^2 \geq 0$. Prin urmare valorile proprii sunt ambele reale şi egale cu $\lambda_1 = \dfrac{\alpha+\beta+\sqrt{\delta}}{2}$ şi $\lambda_2 = \dfrac{\alpha+\beta-\sqrt{\delta}}{2}$. Vectorii proprii pot fi calculaţi prin rezolvarea sistemelor liniare singulare omogene $Ax_1 = \lambda_1 x_1$ and $Ax_2 = \lambda_1 x_2$. Sistemul $Ax_1 = \lambda_1 x_1$ poate fi scris

$$\begin{cases} (\lambda_1 - \alpha)x_{11} - \gamma x_{21} = 0 \\ -\gamma x_{11} + (\lambda_1 - \beta)x_{21} = 0 \end{cases}$$

Dacă $\gamma = 0$, atunci $\lambda_1 = \alpha$ şi o soluţie nenulă a sistemului de mai sus este $x_1 = \begin{bmatrix} \mu \\ 0 \end{bmatrix}$ cu $\mu \neq 0$ arbitrar. Dacă $\gamma \neq 0$, atunci considerând $x_{11} = \mu \neq 0$, o soluţie nenulă a sistemului de mai sus este $x_1 = \begin{bmatrix} \mu \\ \dfrac{(\lambda_1-\alpha)\mu}{\gamma} \end{bmatrix}$. Sistemul $Ax_2 = \lambda_1 x_2$ poate fi scris similar

$$\begin{cases} (\lambda_2 - \alpha)x_{21} - \gamma x_{22} = 0 \\ -\gamma x_{21} + (\lambda_2 - \beta)x_{22} = 0 \end{cases}$$

Dacă $\gamma = 0$, atunci $\lambda_2 = \beta$ si o soluţie nenulă a sistemului de mai sus este $x_2 = \begin{bmatrix} 0 \\ \nu \end{bmatrix}$ cu $\nu \neq 0$ arbitrar. Dacă $\gamma \neq 0$, atunci considerând $x_{22} = \nu \neq 0$, o soluţie nenulă a sistemului

de mai sus este $x_2 = \begin{bmatrix} \frac{(\lambda_2-\beta)\nu}{\gamma} \\ \nu \end{bmatrix}$. Dacă $\gamma = 0$, cei doi vectori proprii sunt evident ortogonali. Dacă $\gamma \neq 0$, atunci

$$x_2^T x_1 = \begin{bmatrix} \frac{(\lambda_2-\beta)\nu}{\gamma} & \nu \end{bmatrix} \begin{bmatrix} \mu \\ \frac{(\lambda_1-\alpha)\mu}{\gamma} \end{bmatrix} = \frac{(\lambda_2-\beta)\nu}{\gamma}\mu + \nu\frac{(\lambda_1-\alpha)\mu}{\gamma} =$$

$$= \frac{\mu\nu}{\gamma}(\lambda_1 - \alpha + \lambda_2 - \beta) = 0.$$

Deci, cei doi vectori proprii sunt ortogonali.

**b.** Dacă $A \in \mathbb{R}^{n\times n}$ este simetrică, i.e. $A^T = A$, fie $S = Q^T AQ$ forma sa Schur reală, care este o structură cvasi-triunghiulară. Avem $S^T = Q^T A^T Q = Q^T AQ = S$, deci matricea $S$ este simetrică şi, prin urmare, are o structură cvasi-diagonală $S = \text{diag}(S_{11}, S_{22}, \ldots, S_{pp})$ cu blocurile diagonale $2\times 2$ având valori proprii complex conjugate. Dar blocurile diagonale sunt de asemenea simetrice şi am văzut mai sus că o matrice $2 \times 2$ simetrică are valori proprii reale. Astfel că toate blocurile diagonale ale unei forme Schur reale a unei matrice simetrice sunt $1 \times 1$, i.e. forma Schur reală a unei matrice simetrice este diagonală (în acest caz forma canonică Jordan şi forma Schur sunt identice), toate valorile proprii sunt reale şi din $S = Q^T AQ = \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ avem $AQ = Q\Lambda$, i.e. $AQ(:, j) = Q\Lambda(:, j) = \lambda_j Qe_j = \lambda_j Q(:, j)$, $j = 1 : n$. Prin urmare, $x_j = Q(:, j)$ sunt vectori proprii ai matricei simetrice $A$, asociaţi valorilor proprii $\lambda_j$ şi, fiind coloane ale unei matrice ortogonale, sunt toţi ortogonali doi câte doi. Aceste proprietăţi utile aduc simplificări importante *algoritmului QR simetric*. Astfel matricea superior Hessenberg obţinută în prima etapă a algoritmului QR este simetrică şi prin urmare *tridiagonală*. De asemenea, toate matricele din secvenţa QR sunt tridiagonale şi convergenţa la forma diagonală este foarte rapidă. Pentru mai multe detalii consultaţi cursul şi bibliografia.

**Problema 4.8** Elaboraţi un algoritm pentru reducerea unei matrice $A \in \mathbf{R}^{n\times n}$ la forma superior Hessenberg $H = TAT^{-1}$, unde $T$ este un produs de transformări gaussiene elementare stabilizate $M_k P_k$ (de tipul celor întâlnite în algoritmul GPP).

**Rezolvare.** Ideea este aceeaşi ca şi în cazul algoritmului de reducere ortogonală **HQ** şi se bazează pe faptul că matricele de permutare $P_k = P_{ki_k}$, $i_k \geq k$ şi matricele elementare inferior triunghiulare $M_k = I_n - m_k e_k^T$ au structura $\begin{bmatrix} I_{k-1} & 0 \\ 0 & X \end{bmatrix}$ la fel cu a matricelor $M_k P_k$, $P_k^{-1} = P_k^T$, $M_k^{-1} = I_n + m_k e_k^T$ şi $P_k^{-1} M_k^{-1}$. Prin urmare, forma Hessenberg poate fi obţinută prin următoarele transformări de asemănare

$$A \leftarrow H = TAT^{-1} = M_{n-1}P_{n-1}\cdots M_3 P_3 M_2 P_2 A (M_2 P_2)^{-1}(M_3 P_3)^{-1}\cdots(M_{n-1}P_{n-1})^{-1} =$$

$$= M_{n-1}P_{n-1}\cdots M_3 P_3 M_2 P_2 A P_2 M_2^{-1} P_3 M_3^{-1}\cdots P_{n-1}M_{n-1}^{-1}.$$

Luând în calcul faptul că o matrice de permutare $P_{ki_k}$ este definită complet de cei doi întregi $(k, i_k)$ putem scrie următoarea schemă de calcul

1. **pentru** $k = 1 : n - 2$

1. Se determină $i_{k+1}$ astfel încât $|a_{i_{k+1}k}| = \max_{i=k+1:n}(|a_{ik}|)$
2. $A(i_{k+1}, k:n) \leftrightarrow A(k+1, k:n)$
3. Se determină matricea elementară inferior triunghiulară $M_{k+1}$ astfel încât $(M_{k+1}A)(k+2:n, k) = 0$
4. $A = M_{k+1}A$
5. $A(:, k+1) \leftrightarrow A(:, i_{k+1})$
6. $A = AM_{k+1}^{-1}$.

Folosind cunoştinţele de utilizare a transformărilor gaussiene, schema de calcul de mai sus este detaliată în următorul algoritm.

1. **pentru** $k = 1 : n-2$
   1. $i_{k+1} = k+1$
   2. $\nu = |a_{k+1,k}|$
   3. **pentru** $i = k+2 : n$
      1. **dacă** $|a_{ik}| > \nu$
         1. $i_{k+1} = i$
         2. $\nu = |a_{ik}|$
   4. **pentru** $j = k : n$
      1. $a_{i_{k+1},j} \leftrightarrow a_{k+1,j}$
   5. **pentru** $i = k+2 : n$
      1. $\mu_{i,k+1} = a_{ik}/a_{k+1,k}$
      2. $a_{ik} = 0$
   6. **pentru** $j = k+1 : n$
      1. **pentru** $i = k+2 : n$
         1. $a_{ij} \leftarrow a_{ij} - \mu_{i,k+1}a_{k+1,j}$
   7. **pentru** $i = 1 : n$
      1. $a_{i_{k+1},k+1} \leftrightarrow a_{i,k+1}$
   8. **pentru** $j = k+1 : n$
      1. **pentru** $i = k+2 : n$
         1. $a_{ij} \leftarrow a_{ij} - \mu_{i,k+1}a_{k+1,j}$

Soluţia este de două ori mai eficientă decât algoritmul **HQ**, dar acesta din urmă este preferat datorită avantajelor transformărilor ortogonale.

**Problema 4.9** Fie $H \in \mathbf{R}^{n\times n}$ o matrice superior Hessenberg. Descrieţi detaliile unui pas QR simplu cu deplasare explicită folosind valoarea optimă $\mu = h_{nn}$ a deplasării.

**Rezolvare.** Fiind dată matricea $H$ curentă din şirul QR, calculul $H \leftarrow H'$ al succesoarei ei reprezintă un pas simplu QR. Versiunea cu deplasare explicită este definitaa prin

$$\begin{cases} [\,Q, R\,] = \mathbf{qr}(H - \mu I_n) & \%\text{factorizareaQR} \\ H \leftarrow H' = RQ + \mu I_n \end{cases}$$

Matricea $H - \mu I$ este superior Hessenberg; factorizarea sa QR poate fi calculată folosind o secvenţă de rotaţii Givens pentru triangularizarea ortogonală a matricei $H - \mu I$, i.e. $P_{n-1,n}\cdots P_{23}P_{12}(H - \mu I) = R$; atunci $Q = P_{12}^T P_{23}^T \cdots P_{n-1,n}^T$. Înmulţirea matricelor $RQ$ se poate face fără calculul explicit al lui $Q$, folosind expresia $RQ = RP_{12}^T P_{23}^T \ldots P_{n-1,n}^T$. Toate calculele pot fi făcute prin suprascrierea matricei $H$. Schema de calcul pentru pasul QR simplu cu deplasare explicită este:

1. $\mu \leftarrow h_{nn}$
2. $H \leftarrow H - \mu I$
3. **pentru** $k = 1 : n-1$
   1. Se calculează rotaţia Givens $P_{k,k+1}$ astfel încât $(P_{k,k+1}H)(k+1,k) = 0$
   2. $H \leftarrow P_{k,k+1}H$
3. **pentru** $k = 1 : n-1$
   2. $H \leftarrow HP_{k,k+1}^T$
5. $H \leftarrow H + \mu I$

Notând cu $(c_k, s_k)$ scalarii definitorii pentru rotaţia $P_{k,k+1}$, schema de mai sus poate fi detaliată în următorul algoritm.

1. $\mu \leftarrow h_{nn}$
2. **pentru** $i = 1 : n$
   1. $h_{ii} \leftarrow h_{ii} - \mu$
3. **pentru** $k = 1 : n-1$
   1. $\rho = \sqrt{h_{kk}^2 + h_{k+1,k}^2}$
   2. $c_k = h_{k,k}/\rho$
   3. $s_k = h_{k+1,k}/\rho$
   4. $h_{k,k} = \rho$
   5. $h_{k+1,k} = 0$
   6. **pentru** $j = k+1 : n$
      1. $\tau = h_{kj}$
      2. $h_{kj} \leftarrow c_k\tau + s_k h_{k+1,j}$
      3. $h_{k+1,j} \leftarrow -s_k\tau + c_k h_{k+1,j}$
4. **pentru** $k = 1 : n-1$
   1. **pentru** $i = 1 : k+1$
      1. $\tau = h_{ik}$
      2. $h_{ik} \leftarrow \tau c_k + h_{i,k+1}s_k$
      3. $h_{i,k+1} \leftarrow -\tau s_k + h_{i,k+1}c_k$
5. **pentru** $i = 1 : n$
   1. $h_{ii} \leftarrow h_{ii} + \mu$

**Problema 4.10** Elaboraţi un algoritm pentru calculul valorilor proprii ale matricei $A = I_n + uv^T$, unde $u, v \in \mathbf{R}^n$.

**Rezolvare.** Dacă $Q = Q^T$ este reflectorul Householder pentru care $Qu = \|u\|e_1$, atunci matricea

$$A' = Q^T AQ = I + \|u\|e_1 v^T Q^T = I + e_1 w^T,$$

unde $w = \|u\|Qv$, este ortogonal asemenea cu $A$ şi superior triunghiulară. Prin urmare valorile proprii ale lui $A$ sunt elementele diagonale ale lui $A'$. Avem $a'_{11} = 1 + w_1 = 1 + \|u\|v^T q_1$, unde $q_1 = Qe_1$ este prima coloană a lui $Q$ şi $a'_{ii} = 1$ pentru $i \geq 2$. Dar $q_1 = u/\|u\|$, prin urmare $a'_{11} = 1 + v^T u$. Astfel, $\lambda_1 = 1 + v^T u$ şi $\lambda_i = 1$, $i = 2 : n$ este o valoare proprie cu ordinul de multiplicitate $n-1$.

**Problema 4.11 a.** Fiind dat un vector $x \in \mathbb{R}^n$ nenul, scrieţi un algoritm pentru calculul unui vector $v \in \mathbb{R}^n$ astfel încât $v^T x = 1$.

**b.** Presupunem că matricea $A \in \mathbb{C}^{n\times n}$ are valorile proprii reale $\lambda_i$, $i = 1 : n$, şi că $x_i$, $i = 1 : n$, sunt vectori proprii asociaţi. Fie $v \in \mathbb{R}^n$ un vector astfel încât $v^T x_1 = 1$ şi considerăm matricea $B = (I_n - x_1 v^T)A$. Arătaţi că $\lambda(B) = \{\, 0, \lambda_2, \ldots, \lambda_n \,\}$, şi că vectorii $x_1^B = x_1$, $x_i^B = x_i - (v_1^T x_i)x_1$ sunt vectori proprii ai matricei $B$.

**Rezolvare. a.** Fie $U_1 = I_n - uu^T/\beta$ un reflector Householder astfel încât $U_1 x = \rho e_1$, $\rho \neq 0$. Atunci

$$v = \frac{1}{\bar{\rho}} U_1 e_1 = \frac{1}{\bar{\rho}}(e_1 - u\tau)$$

unde $\tau = u_1/\beta$, este vectorul pe care îl căutăm. Într-adevăr, $v^T x = \dfrac{1}{\bar{\rho}} e_1^T U_1 x = 1$. Vectorul $v$ poate fi calculat după cum urmează:

1. $\rho = -\sqrt{\sum_{i=1}^n x_i^2}$
2. $u_1 = \rho + x_1$
3. **pentru** $i = 2 : n$
    1. $u_i = x_i$
4. $\beta = u_1 \rho$
5. $\tau = u_1 \beta$
6. $v_1 = (1 - u_1\tau)/\rho$
6. **pentru** $i = 2 : n$
    1. $v_i = -u_i\tau/\rho$

**b.** Avem $Bx_1 = (I_n - x_1 v^T)Ax_1 = \lambda_1(I_n - x_1 v^T)x_1 = 0$; deci $x_1$ este un vector propriu al lui $B$ asociat unei valori proprii nule a lui $B$. De asemenea $Bx_i^B = (I_n - x_1 v^T)A(x_i - (v_1^T x_i)x_1) = (I_n - x_1 v^T)(\lambda_i x_i - \lambda_1 x_1 (v_1^T x_i)) = \lambda_i x_i^B$, $i = 2 : n$.

**Problema 4.12** Scrieţi un algoritm pentru calculul polinomului caracteristic al unei matrice pătrate $A \in \mathbb{R}^{n\times n}$ date.

**Rezolvare.** Cea mai bună metodă numerică actuală constă în calculul iniţial al valorilor proprii $\lambda_i$, folosind algoritmul QR, iar apoi în calculul $p(\lambda) = \prod_{i=1}^n (\lambda - \lambda_i)$. Pentru a determina coeficienţii lui $p(\lambda)$, folosim schema de calcul

1. $p(\lambda) = 1$
2. **for** $i = 1 : n$
    1. $p(\lambda) = p(\lambda)(\lambda - \lambda_i)$

Dacă, la începutul pasului $i$ avem

$$p(\lambda) = p^{(i-1)}(\lambda) = \lambda^{i-1} + p_2^{(i-1)}\lambda^{i-2} + \ldots + p_{i-1}^{(i-1)}\lambda + p_i^{(i-1)}$$

procesarea la pasul $i$ duce la

$$p(\lambda) = p^{(i)}(\lambda) = \lambda^i + (p_2^{(i-1)} - 1 \cdot \lambda_i)\lambda^{i-1} + \ldots + (p_{i-1}^{(i)} - p_i^{(i-1)}\lambda_i)\lambda + (0 - p_i^{(i-1)}\lambda_i) =$$

$$= \lambda^{i-1} + p_2^{(i)}\lambda^{i-2} + \ldots + p_i^{(i)}\lambda + p_{i+1}^{(i)}.$$

Prin urmare, $p_1^{(i)} = p_1^{(i-1)} = 1$ şi $p_j^{(i)} = p_j^{(i-1)} - p_{j+1}^{(i-1)}\lambda_i$ pentru $j \geq 2$. Calculând coeficienţii în ordinea inversă putem suprascrie coeficienţii de la pasul anterior precum în următorul algoritm. Astfel, cel mai bun algoritm pentru calculul coeficienţilor polinomului caracteristic este:

1. Folosind algoritmul QR, calculaţi $\lambda_i$, $i = 1 : n$
2. $p_1 = 1$
3. **pentru** $i = 2 : n + 1$
   1. $p_i = 0$
4. **pentru** $i = 1 : n$
   1. **pentru** $j = i + 1 : -1 : 2$
      1. $p_j \leftarrow p_j - \lambda_i p_{j-1}$.

Evident, efortul principal de calcul constă în determinarea valorilor proprii.

## 4.3 Probleme propuse

**Problema 4.13** Fie date matricele

$$A = \begin{bmatrix} 3 & -3 & 2 \\ -1 & 5 & -2 \\ -1 & 3 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} -4 & 0 & 8 \\ -8 & 3 & 9 \\ -4 & -1 & 9 \end{bmatrix}.$$

Folosind definiţiile valorilor şi vectorilor proprii calculaţi valorile şi vectorii proprii pentru $A$ şi $B$. După aceea calculaţi-le şi formele Schur.

**Problema 4.14 a.** O matrice pătrată se numeşte *nilpotentă* dacă există un număr întreg $k \geq 0$ astfel încât $A^k = 0$. Demonstraţi că toate valorile proprii ale unei matrice nilpotente sunt nule. Daţi un exemplu de matrice $2 \times 2$ nilpotentă.

**b.** O matrice pătrată se numeşte *idempotentă* dacă $A^2 = A$. Demonstraţi că toate valorile proprii ale unei matrice idempotente sunt 0 sau 1. Daţi un exemplu de matrice $2 \times 2$ idempotentă care să nu fie $I_2$.

**Problema 4.15 a.** Calculaţi valorile proprii şi un set de vectori proprii pentru un reflector Householder $U_k$. **b.** Calculaţi valorile proprii şi un set de vectori proprii pentru o rotaţie plană $P_{ij}$.

**Problema 4.16** Care va fi rezultatul aplicării metodei puterii matricelor:

$$A = \begin{bmatrix} 5 & 8 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & -1 \\ 1 & 3 \end{bmatrix},$$

cu aproximările iniţiale de rigoare? Ce înseamnă "de rigoare" în aceste cazuri?

**Problema 4.17** Presupunem că avem date matricea $A \in \mathbb{R}^{n\times n}$ şi vectorul $z \in \mathbb{R}^n$. Scrieţi un algoritm pentru calculul unei matrice ortogonale $Q$ astfel încât $Q^TAQ$ să fie superior Hessenberg şi $Q^Tz$ coliniară cu vectorul $e_1$.

**Problema 4.18** Fie dată o pereche (valoare proprie, vector propriu)$= (\lambda, x)$ a unei matrice superior Hessenberg $H \in \mathbb{R}^{n\times n}$. Scrieţi un algoritm pentru calculul unei matrice ortogonale $Q$ astfel încât matricea $Q^THQ$ să aibă structura $Q^THQ = \begin{bmatrix} \lambda & f^T \\ 0 & G \end{bmatrix}$, unde matricea $G \in \mathbb{R}^{(n-1)\times(n-1)}$ este superior Hessenberg.

**Problema 4.19** Fie $H \in \mathbb{R}^{n\times n}$ o matrice superior Hessenberg dată şi următoarea procedură pentru calculul matricei succesoare $H \leftarrow H'$ într-un algoritm iterativ:

1. Se aplică algoritmul GPP $M_{n-1}P_{n-1}\dots M_1P_1H = R$ pentru a obţine matricea superior triunghiulară $R$.
2. $H \leftarrow H' = RP_1M_1^{-1}\dots P_{n-1}1M_{n-1}^{-1}$,

Demonstraţi că:

**a.** matricea succesoare $H'$ este superior Hessenberg şi că,

**b.** matricea succesoare $H'$ este asemenea cu $H$.

**Problema 4.20** **a.** Considerăm matricea $H \in \mathbb{R}^{2\times 2}$ cu spectru real şi fie $H_k \overset{\text{not}}{=} \begin{bmatrix} \alpha & \gamma \\ \epsilon & \beta \end{bmatrix}$ matricea curentă din şirul **QR** pentru matricea $H$. Folosind deplasarea $\mu_k = \beta$ calculaţi matricea succesoare $H_{k+1}$. Examinaţi elementul $H_{k+1}(2,1)$ şi deduceţi unele proprietăţi de convergenţă ale şirului **QR**.

**b.** Considerăm matricea simetrică $T \in \mathbb{R}^{2\times 2}$ şi fie $T_k \overset{\text{not}}{=} \begin{bmatrix} \alpha & \epsilon \\ \epsilon & \beta \end{bmatrix}$ matricea curentă a şirului **QR** pentru matricea $T$. Folosind deplasarea $\mu_k = \beta$ calculaţi matricea succesoare $T_{k+1}$. Examinaţi elementele extradiagonale $T_{k+1}(1,2) = T_{k+1}(2,1)$ şi deduceţi unele proprietăţi de convergenţă ale şirului **QR** simetric .

## 4.4 Bibliografie


1. B. Jora, B. Dumitrescu, C. Oară, Numerical Methods, UPB, Bucharest, 1995.
2. G.W. Stewart, Introduction to Matrix Computations, Academic Press, 1973.
3. G. Golub, Ch. Van Loan, Matrix Computations, 3-rd edition, John Hopkins University Press, 1998.
4. G.W. Stewart, Matrix Algorithms, vol.1: Basic Decompositions, SIAM, 1999.
5. G.W. Stewart, Matrix Algorithms, vol.2: Eigensystems, SIAM, 2001.
6. B. Dumitrescu, C. Popeea, B. Jora, Metode de calcul numeric matriceal. Algoritmi fundamentali, ALL, Bucureşti, 1998.

## 4.5 Programe MATLAB

**Problema 4.5 a**

```
1:  function [lambda]= p45a(A,x)
2:  %--------------------
3:  % Rezolvarea problemei 4.5 subpunctul a).
4:  % Algoritmul calculeaza catul Rayleigh definit de perechea (A,x).
5:  % Daca x este un vector propriu al matricei patratice A, atunci
6:  % lambda este valoarea proprie asociata vectorului propriu x.
7:  % Apel:
8:  %      lambda=p45a(A,x);
9:  % Stamatescu Grigore, mai 2006.
10: %--------------------
11: [m,n]=size(A);
12: % verificam daca matricea A este patratica
13: if m~=n,
14:       error('Matricea A nu este patratica');
15: end
16: sigma=0;
17: tau=0;
18: for i=1:n
19:       y(i)=0;
20:       for j=1:n
21:             y(i)=y(i)+A(i,j)*x(j);
22:       end
23:       sigma=sigma+x(i)*y(i);
24:       tau=tau+x(i)*x(i);
25: end
26: lambda=sigma/tau;
```

```
1:  function [x]= eigenvector(A)
2:  %--------------------
3:  % Rezolvarea problemei 4.5 subpunctul a).
4:  % Algoritmul calculeaza un vector propriu al matricei patratice A, folosind
5:  % functia MATLAB eig, necesar pentru testarea programului precedent.
6:  % Stamatescu Grigore, mai 2006.
7:  %--------------------
8:  [m,n]=size(A);
9:  % verificam daca A este patratica
10: if m~=n,
11:       error('Matricea A nu este patratica');
12: end
13: [V,D]=eig(A);
14: x=V(1:n);
```

**Problema 4.5 b**

```
1:  function [x]= p45b(A,lambda)
2:  %-------------------------
3:  % Algoritmul calculeaza vectorul propriu x al unei matrice A atunci
4:  % cand valoarea proprie lambda corespunzatoare este cunoscuta.
5:  % Apelul:
6:  %       x=p45b(A,lambda)
7:  % Stamatescu Grigore, mai 2006.
8:  %-------------------------
9:  [m,n]=size(A);
10: % verificam daca matricea A este patratica
11: if m~=n,
12:       error('Matricea A nu este patratica');
13: end
14: T=lambda*eye(size(A))-A;
15: [T,M,p]=GPP(T);
16: k=1;
17: for i=1:n-1
18:       if A(i,i)~=0,
19:             k=k+1;
20:       end
21: end
22:
23: for i=k+1:n
24:       x(i)=0;
25: end
26: x(k)=1;
27:
28: if k>1,
29:       for i=1:k-1
30:             x(i)=UTRIS(T(i,i),-T(i,k));
31:       end
32: end
```

**Problema 4.6**

```
1:  function [A]= Schr(A)
2:  %-------------------------
3:  % Algoritmul calculeaza o forma Schur a matricei patratice A cu spectru real
4:  % folosind o procedura de calcul a unui vector propriu.
5:  % Apel:
6:  %       A=Schr(A);
7:  % Stamatescu Grigore, mai 2006.
8:  %-------------------------
9:  [m,n]=size(A);
10: if m~=n,
```

```
11:         error('Matricea A nu este patratica');
12:     end
13:     [V,D]=eig(A); % ineficient!
14:     for t=1:n
15:         if isreal(D(t,t))~=1,
16:             error('Matricea A nu are spectrul valorilor proprii real!');
17:         end
18:     end
19:
20:     for k=1:n-1
21:         x=eigenvector(A(k:n,k:n));
22:         suma=0;
23:         for i=1:n-k+1
24:             suma=suma+x(i)*x(i);
25:         end
26:         alfa=sqrt(suma);
27:         for i=1:n-k+1
28:             x(i)=x(i)/alfa;
29:         end
30:         u(1)=x(1)-1;
31:         for i=2:n-k+1
32:             u(i)=x(i);
33:         end
34:         beta=-u(1);
35:         for j=k:n
36:             suma1=0;
37:             for i=k:n
38:                 suma1=suma1+u(i-k+1)*A(i,j);
39:             end
40:             tau=suma1/beta;
41:             for i=k:n
42:                 A(i,j)=A(i,j)-tau*u(i-k+1);
43:             end
44:         end
45:         for i=1:n
46:             suma2=0;
47:             for j=k:n
48:                 suma2=suma2+A(i,j)*u(j-k+1);
49:             end
50:             tau=suma2/beta;
51:             for j=k:n
52:                 A(i,j)=A(i,j)-tau*u(j-k+1);
53:             end
54:         end
55:     end
```

**Problema 4.8**

```
1:  function [A]= Hessenberg(A)
2:  %------------------------
3:  % Algoritmul reduce matricea A patratica la forma Hessenberg.
4:  % Apel:
5:  %      A=Hessenberg(A)
6:  % Stamatescu Grigore mai 2006
7:  %------------------------
8:  [m,n]=size(A);
9:  % verificam daca A este patratica
10: if m~=n,
11:       error('Matricea A nu este patratica');
12: end
13:
14: for k=1:n-2
15:       z(k+1)=k+1;
16:       niu=abs(A(k+1,k));
17:       for i=k+2:n
18:             if abs(A(i,k))>niu,
19:                   z(k+1)=i;
20:                   niu=abs(A(i,k));
21:             end
22:       end
23:       for j=k:n
24:             A(z(k+1),j)=A(k+1,j);
25:       end
26:       for i=k+2:n
27:             miu(i,k+1)=A(i,k)/A(k+1,k);
28:             A(i,k)=0;
29:       end
30:       for j=k+1:n
31:             for i=k+2:n
32:                   A(i,j)=A(i,j)-miu(i,k+1)*A(k+1,j);
33:             end
34:       end
35: for i=1:n
36:             A(z(k+1),k+1)=A(i,k+1);
37: end
38:       for j=k+1:n
39:             for i=k+2:n
40:                   A(i,j)=A(i,j)-miu(i,k+1)*A(k+1,j);
41:             end
42:       end
43: end
```

**Problema 4.9**

```
1:  function [H]= pasQR(H)
2:  %-------------------------
3:  % Fiind data o matrice H superior Hessenberg algoritmul implementeaza
4:  % un pas QR cu deplasare explicita folosind deplasarea miu=H(n,n)
5:  % Apel:
6:  %      H=pasQR(H)
7:  % Stamatescu Grigore, mai 2006.
8:  %-------------------------
9:  [m,n]=size(H);
10: % verificam daca H este patratica
11: if m~=n,
12:       error('Matricea H nu este patratica');
13: end
14: % analizam daca matricea este superior Hessenberg
15: for i=3:n
16: for j=1:i-2
17:             if H(i,j)~=0
18:                   error('Matricea H nu e superior Hessenberg')
19:             end
20:       end
21: end
22: miu=H(n,n);
23: for i=1:n
24:       H(i,i)=H(i,i)-miu;
25: end
26: for k=1:n-1
27:       ro=sqrt(H(k,k)*H(k,k)+H(k+1,k)*H(k+1,k));
28:       c(k)=H(k,k)/ro;
29:       s(k)=H(k+1,k)/ro;
30:       H(k,k)=ro;
31:       H(k+1,k)=0;
32:       for j=k+1:n
33:             tau=H(k,j);
34:             H(k,j)=c(k)*tau+s(k)*H(k+1,j);
35:             H(k+1,j)=-s(k)*tau + c(k)*H(k+1,j);
36:       end
37: end
38: for k=1:n-1
39:       for i=1:k+1
40:             tau=H(i,k);
41:             H(i,k)=tau*c(k)+H(i,k+1)*s(k);
42:             H(i,k+1)=-tau*s(k)+H(i,k+1)*c(k);
43:       end
44: end
45: for i=1:n
46:       H(i,i)=H(i,i)+miu;
47: end
```

**Problema 4.11**

```
1:  function [v]= vtx1(x)
2:  %-------------------------
3:  % Fiind dat un vector x nenul de dimensiune n, algoritmul calculeaza
4:  % un vector v (dimensiune n) astfel incat v^T*x=1
5:  % Apel:
6:  %       v=vtx1(x)
7:  % Stamatescu Grigore, mai 2006.
8:  %-------------------------
9:  n=length(x);f=0;
10: % verificam daca x este nenul
11: for i=1:n
12:       if x(i)==0,
13:             f=f+1;
14:       end
15: end
16: if f==n,
17:       error('Vectorul x este nul!');
18: end
19: s=0;
20: for i=1:n
21:       s=s+x(i)*x(i);
22: end
23: ro=-sqrt(s);
24: u(1)=ro+x(1);
25: for i=2:n
26:       u(i)=x(i);
27: end
28: beta=u(1)*ro;
29: tau=u(1)*beta;
30: v(1)=(1-u(1)*tau)/ro;
31: for i=2:n
32:       v(i)=-u(i)*tau/ro;
33: end
```

**Problema 4.12**

```
1:  function [p]= polinom(A)
2:  %-------------------------
3:  % Algoritmul calculeaza polinomul caracteristic p al matricei patratice
4:  % date A (nxn).
5:  % Apel:
6:  %       p=polinom(A)
7:  % Stamatescu Grigore, mai 2006.
```

```
8:  %-----------------------
9:  [n,m]=size(A);
10: % analizam daca matricea e patratica
11: if n~=m
12:     error('Matricea nu e patratica');
13: end
14:
15: [V,D]=eig(A);
16: for i=1:n
17:     lambda(i)=D(i,i);
18: end
19: p(1)=1;
20: for i=2:n+1
21:     p(i)=0;
22: end
23: for i=1:n
24:     for j=i+1:-1:2
25:         p(j)=p(j)-lambda(i)*p(j-1);
26:     end
27: end
```