Lucrarea de laborator nr. 13

I. Scopul lucrării

Valori şi vectori proprii

II. Conținutul lucrării

1. Valori şi vectori proprii. Polinom caracteristic. Noțiuni generale.
2. Comenzi MAPLE pentru calculul valorilor şi vectorilor proprii.
3. Calculul valorilor proprii ale unei matrice simetrice tridiagonale. Calculul valorilor proprii ale unei matrice simetrice.
4. Calculul vectorilor şi valorilor proprii pentru matrice oarecare. Metoda puterii. Algoritmul QR

III. Prezentarea lucrării

III.1. Valori şi vectori proprii. Polinom caracteristic. Noțiuni generale.

Fie A o matrice cu n linii şi n coloane cu coeficienții din corpul K, unde K este **R** sau **C**. Scalarul λ din K se numeşte ***valoare proprie*** pentru A dacă există vectorul $x \in K^n$ astfel încât:

$$Ax = \lambda x$$

Vectorul x se numeşte ***vector propriu*** asociat valorii proprii λ. Relația $Ax = \lambda x$ poate fi scrisă echivalent $(\lambda I_n - A)x = 0$, unde I_n este matricea unitatea de ordinul n. Sistemul liniar omogen $(\lambda I_n - A)x = 0$ admite o soluție nenulă x dacă şi numai dacă $\det(\lambda I_n - A) = 0$. Polinomul $P_A(\lambda) = \det(\lambda I_n - A)$ se numeşte ***polinom caracteristic*** al matricei A. Orice matrice cu coeficienți complecşi are exact n valori proprii (reale sau complexe, nu neapărat distincte). Acestea sunt rădăcinile polinomului caracteristic. Rădăcinile polinoamelor de grad mai mare decât strict ca patru nu pot fi întotdeauna calculate într-un număr finit de paşi. Astfel calculul valorilor proprii se face iterativ. Calculul

valorilor proprii ca rădăcini ale polinomului caracteristic nu este de obicei recomandabil din cauza

- volumului de calcul necesar pentru găsirea coeficienților polinomului
- volumului de calcul necesar găsirii rădăcinilor polinomului

Considerăm un exemplu din care va rezulta că şi în situația $n \leq 4$ găsirea valorilor proprii ca rădăcini ale polinomului caracteristic poate să nu fie recomandabilă (se presupune că se lucrează în virgulă mobilă). Fie ε un număr pozitiv mai mare decât cel mai mic număr în virgulă mobilă reprezentabil în calculator într-o anumită precizie. Presupunem că ε^2 este mai mic decât cel mai mic număr reprezentabil, şi deci va fi reprezentat ca fiind 0. Fie matricea

$$A = \begin{pmatrix} 1 & \varepsilon \\ \varepsilon & 1 \end{pmatrix}$$

ale cărei valori proprii sunt $1 + \varepsilon$ şi $1 - \varepsilon$, deoarece polinomul caracteristic este

$$P_A(\lambda) = (\lambda - 1)^2 - \varepsilon^2 = \lambda^2 - 2\lambda + 1 - \varepsilon^2$$

În virgulă mobilă însă

$$P_A(\lambda) = \lambda^2 - 2\lambda + 1$$

şi în consecință valorile proprii determinate sunt 1 şi 1, diferite de $1 + \varepsilon$ şi $1 - \varepsilon$ în precizia considerată.

Multiplicitatea (algebrică) valorii proprii λ este dată de multiplicitatea ei ca rădăcină a polinomului caracteristic. Vectorii proprii asociați valorii proprii λ formează un subspațiu vectorial al lui K^n invariant la A (dacă S este subspațiul vectorilor proprii asociați lui λ atunci $AS \subset S$). Dimensiunea subspațiului vectorilor proprii asociați lui λ se numeşte ***multiplicitatea geometrică*** a lui λ.

Dacă λ este valoare proprie pentru A şi α este un scalar oarecare, atunci $\lambda - \alpha$ este valoare proprie pentru $A - \alpha I_n$.

Dacă matricea A este inversabilă (nesingulară), şi $Ax = \lambda x$ cu $x \neq 0$, atunci $\lambda \neq 0$ şi $A^{-1}x = \frac{1}{\lambda}x$. Deci valorile proprii ale inversei lui A sunt inversele valorilor proprii ale lui A.

Dacă $Ax = \lambda x$, atunci $A^k x = \lambda^k x$. Astfel valorile proprii ale puterii k a matricei A sunt puterile k ale valorilor proprii ale matricei A. Mai general, dacă p este un polinom şi $Ax = \lambda x$, atunci $p(A)x = p(\lambda)x$.

Dacă A şi B sunt două matrice similare (A şi B sunt similare dacă şi numai dacă există o matrice nesingulară T astfel încât $B = T^{-1}AT$), atunci ATx

= λTx dacă şi numai dacă Bx = λx. Deci A şi B au aceleaşi valori proprii şi dacă x este un vector propriu pentru B, atunci Tx este un vector propriu pentru A. Folosind această observaţie, pentru determinarea valorilor şi vectorilor proprii asociaţi lui A se determină o matrice B similară cu A cu formă convenabilă. Dacă B are formă diagonală sau mai general triunghiulară, atunci elementele de pe diagonala principală reprezintă valorile proprii.

III. 2. Comenzi MAPLE pentru calculul valorilor şi vectorilor proprii

Comenzile pentru calculul valorilor şi vectorilor proprii sunt incluse în pachetul linalg. Comanda

```
>charmat(A, lambda);
```

întoarce matricea caracteristică asociată lui A, adică matricea $\lambda I_n - A$.

Comanda

```
>charpoly(A, lambda);
```

întoarce polinomul caracteristic asociat lui A, adică $\det(\lambda I_n - A)$. Parametrul lambda de pe a doua poziţie indică numele folosit pentru nedeterminata polinomului.

Comanda

```
>eigenvals(A);
```

întoarce secvenţa valorilor proprii asociate lui A. Dacă unul dintre coeficienţii matricei A este în virgulă mobilă, atunci este utilizată o metodă numerică pentru determinarea valorilor proprii, precizia cu care se lucrează fiind cea stabilită de Digits. În cazul în care coeficienţii matricei A sunt întregi sau raţionali (cazul simbolic) valorile proprii sunt calculate ca rădăcini ale polinomului caracteristic. Dacă un al doilea parametru (opţional) 'implicit' este menţionat valorile proprii sunt indicate folosind notaţia RootOF. Implicit sau în cazul în care se utilizează parametru 'radical' valorile proprii sunt indicate în termeni de radicali exacţi. Dacă gradul polinomului caracteristic este mai mare decât patru acest lucru nu este întotdeauna posibil.

Comanda

```
> Eigenvals(A);
```

întoarce un tablou ce conţine valorile proprii ale lui A. În general, valorile proprii sunt calculate utilizând algoritmul QR. Dacă A este simetrică se utilizează un algoritm mai rapid. Dacă se utilizează un parametru opţional vecs (adică se utilizează comanda sub forma Eigenvals(A,vecs)), atunci numele vecs este folosit pentru o matrice cu n linii şi n coloane ce conţine vectori proprii asociaţi lui A. Coloana i din matricea vecs este un vector propriu asociat celei de-a i-a valoarea proprie. Numele vecs trebuie să fie

neasignat în momentul utilizării, în caz contrar interpretarea comenzii fiind diferită. Comanda Eigenvals în sine este o comandă inertă, pentru calculul efectiv al vectorilor şi valorilor proprii ale lui A, utilizatorul trebuie să folosească evalf (comanda pentru evaluare în virgulă mobilă).

Comanda

```
> eigenvects(A);
```

întoarce vectorii şi valorile proprii asociate lui A. Rezultatul comenzii este o secvență de liste de forma [ei,mi,{v[1,i],…,v[ni,i]}], unde ei este a i-a valoare proprie, mi este multiplicitatea ei algebrică, iar este o mulțime liniar independentă ce generează spațiul de vectorilor proprii asociați valorii proprii ei. Ca şi în cazul comenzii eigenvals dacă un coeficient al lui A este în virgulă mobilă, atunci calculele se efectuează în virgulă mobilă. În caz contrar se utilizează calculul exact (simbolic): se calculează valorile proprii ca rădăcini ale polinomului caracteristic, şi pentru fiecare valoare proprie λ se calculează subspațiul nul al matricei caracteristice λI_n-A.

Exemple

```
>with(linalg);
> A := matrix(3,3,[1,2,3,1,2,3,1,5,6]);
```

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 5 & 6 \end{pmatrix}$$

```
> charmat(A,lambda);
```

$$\begin{pmatrix} \lambda - 1 & -2 & -3 \\ -1 & \lambda - 2 & -3 \\ -1 & -5 & \lambda - 6 \end{pmatrix}$$

```
> charpoly(A,lambda);
```

$$\lambda^3 - 9\,\lambda^2$$

```
> eigenvals(A);
                        0, 0, 9
> A := matrix(3,3,
[1.0,2.0,3.0,1.0,2.0,3.0,2.0,5.0,6.0]);
```

$$A := \begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 1.0 & 2.0 & 3.0 \\ 2.0 & 5.0 & 6.0 \end{pmatrix}$$

```
> eigenvals(A);
   9.321825380, .7700622123 10^-15   , -.3218253805
```

```
>  A:= matrix(3,3, [1,2,3,1,2,3,2,5,6]);
```

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 2 & 5 & 6 \end{pmatrix}$$

```
> eigenvals(A);
```

$$0,\ \frac{9}{2}+\frac{1}{2}\sqrt{93},\ \frac{9}{2}-\frac{1}{2}\sqrt{93}$$

```
>  A:= matrix(5,5,
[1,2,3,1,2,3,2,5,6,7,8,1,2,3,4,9,7,8,2,9,7,6,4,1,2];
```

$$A := \begin{pmatrix} 1 & 2 & 3 & 1 & 2 \\ 3 & 2 & 5 & 6 & 7 \\ 8 & 1 & 2 & 3 & 4 \\ 9 & 7 & 8 & 2 & 9 \\ 7 & 6 & 4 & 1 & 2 \end{pmatrix}$$

```
> eigenvals(A);
 RootOf(975 - 392 _Z - 588 _Z^2  - 159 _Z^3  - 9 _Z^4  +
                          _Z^5 )

>  A:= matrix(5,5, [1.0,2,3,1,2,
3,2,5,6,7,8,1,2,3,4,9,7,8,2,9,7,6,4,1,2]);
```

$$A := \begin{pmatrix} 1.0 & 2 & 3 & 1 & 2 \\ 3 & 2 & 5 & 6 & 7 \\ 8 & 1 & 2 & 3 & 4 \\ 9 & 7 & 8 & 2 & 9 \\ 7 & 6 & 4 & 1 & 2 \end{pmatrix}$$

```
> eigenvals(A);

19.02898145,.9131799116,-3.504777507+ 1.408608557 I,
    -3.504777507 - 1.408608557 I, -3.932606351
> A := array([[1,2,4],[3,7,2],[5,6,9]]);
```

$$A := \begin{pmatrix} 1 & 2 & 4 \\ 3 & 7 & 2 \\ 5 & 6 & 9 \end{pmatrix}$$

```
> B := array([[1,2,3],[1,2,3],[2,5,6]]);
```

$$B := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 2 & 5 & 6 \end{pmatrix}$$

```
> evalf(Eigenvals(A));

      [-.8946025434, 13.74788901, 4.146713483]

> lambda := evalf(Eigenvals(B,vecs));

    λ := [9.321825413, .43 10^(-8)  , -.3218253805]

> print(vecs);
```

$$\begin{pmatrix} -.4286091784 & .9434028232 & .7693450018 \\ -.4286091788 & -.46\ 10^{-9} & .7693450120 \\ -.9031974612 & -.3144676070 & -.8518765905] \end{pmatrix}$$

```
> A := matrix(3,3, [1,-3,3,3,-5,3,6,-6,4]);
```

$$A := \begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix}$$

```
> e := eigenvals(A);
                    e := 4, -2, -2
> v := [eigenvects(A)];
 v := [[-2, 2, {[1, 1, 0], [-1, 0, 1]}], [4, 1, {[1,
                      1, 2]}]]
> v[1][1];
                          4
> v[1][2];
                          1
> v[1][3];
                     {[1, 1, 2]}
> v[2][1];
                         -2
> v[2][2];
                          2
```

III.3. Calculul valorilor proprii ale unei matrice simetrice tridiagonale. Calculul valorilor proprii ale unei matrice simetrice.

Fie A o matrice simetrică şi tridiagonală:

$$A = \begin{pmatrix} a_1 & b_1 & 0 & \dots & 0 & 0 & 0 & 0 \\ b_1 & a_2 & b_2 & \dots & 0 & 0 & 0 & 0 \\ \dots & & & & & & & \\ 0 & 0 & 0 & \dots & 0 & b_{n-2} & a_{n-1} & b_{n-1} \\ 0 & 0 & 0 & \dots & 0 & 0 & b_{n-1} & a_n \end{pmatrix}$$

Presupunem că toate elementele b_i sunt nenule. În caz contrar descompunem matricea A în submatrice simetrice tridiagonale care să îndeplinească această condiție. Pentru fiecare i=1,2,…, n, notăm:

$$A_i = \begin{pmatrix} a_1 & b_1 & 0 & \dots & 0 & 0 & 0 & 0 \\ b_1 & a_2 & b_2 & \dots & 0 & 0 & 0 & 0 \\ \dots & & & & & & & \\ 0 & 0 & 0 & \dots & 0 & b_{i-2} & a_{i-1} & b_{i-1} \\ 0 & 0 & 0 & \dots & 0 & 0 & b_{i-1} & a_i \end{pmatrix}$$

Considerăm polinoamele definite prin:

$$\begin{cases} p_0(\lambda) := 1 \\ p_1(\lambda) := a_1 - \lambda \\ p_i(\lambda) := (a_i - \lambda) p_{i-1}(\lambda) - b_{i-1}^2 p_{i-2}(\lambda),\ 2 \le i \le n \end{cases}$$

Prin dezvoltarea determinantului matricei $\lambda I - A_i$ după ultima linie (sau coloană) se observă că $(-1)^i p_i$ este polinomul caracteristic al matricei A_i. Polinoamele $(p_i)_{1\le i\le n}$ formează un şir Sturm, adică îndeplinesc condițiile:

1) $p_i(\lambda) = 0 \Rightarrow p_{i-1}(\lambda)p_{i+1}(\lambda) < 0$ pentru orice $1 \le i \le n-1$
2) $\lim_{\lambda\to\infty} p_i(\lambda) = +\infty$ pentru orice $1 \le i \le n$.

Ca urmare rădăcinile tuturor polinoamelor din şirul $(p_i)_{1\le i\le n}$ sunt reale şi distincte şi rădăcinile oricărui polinom p_{i+1} sunt separate de rădăcinile polinomului p_i. Pentru fiecare $1 \le i \le n$, notăm

$$s(p_i(\lambda)) = \begin{cases} sign(p_i(\lambda)), \text{ dacă } p_i(\lambda) \neq 0 \\ sign(p_{i-1}(\lambda)), \text{ dacă } p_i(\lambda) = 0 \end{cases}$$

şi N(i, λ) numărul de schimbări de semn între elementele consecutive ale mulţimii:

$$+, s(p_1(\lambda)), s(p_2(\lambda)), \ldots, s(p_i(\lambda))$$

Dacă în matricea simetrică tridiagonală A îndeplineşte condiția $b_i \neq 0$ pentru orice $1 \leq i \leq n-1$, atunci N(i, λ) reprezintă numărul de rădăcini ale polinomului p_i, care sunt mai mici decât λ. Folosind acest fapt putem aproxima orice valoare proprie a matricei A. Presupunem că valorile proprii ale matricei A sunt ordonate crescător: $\lambda_1, \lambda_2, \ldots, \lambda_n$. Pentru a determina valoarea proprie λ_k se începe prin a determina un interval $[a_0, b_0]$ ce conține λ_k, de exemplu $-a_0 = b_0 = \|A\|_\infty$. Apoi luăm $c_0 = \frac{a_0 + b_0}{2}$ calculăm N(n, c_0):

dacă $N(n, c_0) \geq k$, atunci $\lambda_k \in [a_0, c_0]$

dacă $N(n, c_0) < k$, atunci $\lambda_k \in [c_0, b_0]$

Se aplică acelaşi procedeu pentru noul interval obținut.

Algoritm:

Date de intrare:

k – rangul valorii proprii ce urmează a fi calculată

eps: eroarea cu care valoarea proprie λ_k urmează a fi aproximată

Date de ieşire:

c - aproximația celei de a k-a valori proprii

$a = -\|A\|_\infty$

$b = \|A\|_\infty$

cât timp |a-b|≥eps **execută**

c:= (a+b)/2;

dacă $N(n, c_0) \geq k$ **atunci** b: = c

altfel a: =c

Pentru calculul valorilor proprii ale unei matrice simetrice A se poate proceda în felul următor:

1) se determină o matrice B similară cu A, astfel încât B este simetrică şi tridiagonală

2) se calculează valorile proprii ale matricei B (care sunt aceleaşi cu ale matricei A) folosind metoda prezentată mai sus (metoda Givens)

Prezentăm în continuare metoda Householder pentru determinarea matricei B tridiagonală similară cu A. B se obţine prin transformări ortogonale cu matrice de forma $H = I - 2uu^t$ unde $u \in \mathbf{R}^n$ şi $\|u\|_2 = 1$. O astfel de matrice se numeşte matrice Householder. H este simetrică şi $H^2 = I$.

Fie $x = (x_1, x_2, \ldots, x_n)^t \in \mathbf{R}^n$ un vector nenul. Atunci există o matrice Householder H_k astfel încât primele k-1 componente ale vectorului $H_k x$ să coincidă cu primele k-1 componente ale vectorului x, iar ultimele n-k componente ale vectorului $H_k x$ să fie nule, adică

$$H_k x = (x_1, x_2, \ldots, x_{k-1}, \rho_k, 0, 0, \ldots, 0)^t$$

Matricea H_k se obţine ca

$$H_k = I - \frac{2}{\|v\|_2^2} v_k v_k^t ,$$

unde vectorul $v = (0, 0, \ldots 0, v_{kk}, \ldots, v_{nk})^t \in \mathbf{R}^n$ este determinat de

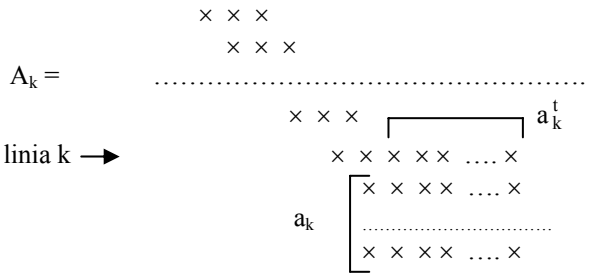
$$\begin{cases} s = \varepsilon \left(\sum_{i=k}^{n} x_i^2 \right)^{1/2} \\ v_{kk} = x_k + s \\ v_{ki} = x_i, \; k+1 \le i \le n \\ \|v\|_2^2 = 2s v_{kk} = 2s(x_k + s) \end{cases}$$

unde $\varepsilon = -1$ dacă $x_k < 0$, şi $\varepsilon = 1$ în caz contrar. Cu aceleaşi notaţii $\rho_k = -s$.

Procedura de tridiagonalizare a matricei A (de obţinere a matricei B) se realizează în n-2 etape. Astfel $A_1 = A$ şi $A_k = H_{k-1}^t A_{k-1} H_{k-1}$, $2 \le k \le n-1$, unde

$$H_{k-1} = \begin{pmatrix} I_k & O \\ O & \widetilde{H}_{k-1} \end{pmatrix}$$

cu $\widetilde{H}_{k-1}$ matrice Householder aleasă astfel încât A_k să aibă forma:



Notăm cu a_{ij}^k, $1\le i,j\le n$ coeficienții matricei A_k şi cu a_{ij}^{k+1}, $1\le i,j\le n$ coeficienții matricei A_{k+1}. Avem $a_{ij}^{k+1}=a_{ij}^k$ pentru orice $1\le i,j\le k$. Pentru ca A_{k+1} să fie tridiagonală trebuie ca $\widetilde{H}_k a_k$ să aibă componentele nule, cu excepția primei componente. Dacă $a_{ik}^k = 0$ pentru orice i = k+2, k+3, …,n, atunci se ia $H_k = I$, altfel matricea A_{k+1} se construieşte cu ajutorul matricei Householder

$$H_k = I - 2\frac{v_k v_k^t}{v_k^t v_k}$$

unde

$$v_k = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ a_{k+1k}^k + \varepsilon\left(\sum_{i=k+1}^{n}\left(a_{ik}^k\right)^2\right)^{1/2} \\ a_{k+2k}^k \\ \vdots \\ a_{nk}^k \end{pmatrix}$$

iar $\varepsilon = -1$ dacă $a_{k+1k}^k < 0$, şi $\varepsilon = 1$ în caz contrar. Dacă notăm

$$w_k = \left(v_k^t v_k\right)^{-1/2} v_k \text{ şi } q_k = 2(I - w_k w_k^t)A_k w_k,$$

atunci

$$A_{k+1} = A_k - w_k q_k^t - q_k w_k^t$$

Deci coeficienții care se modifică sunt dați de formulele:

$$a_{ij}^{k+1} = a_{ij}^k - (w_k)_i (q_k)_j - (w_k)_j (q_k)_i , \quad k+1 \le i,j \le n.$$

Deoarece matricea A_{k+1} este simetrică este suficientă aplicarea acestor formule doar pentru $k+1 \le i \le j \le n$.

Matricea $B = A_{n-2} = Q^t A Q$, unde $Q = Q_1 Q_2 \ldots Q_{n-2}$ este o matrice tridiagonală similară cu A.

Proceduri MAPLE pentru calculul valorilor proprii ale unei matrice simetrice tridiagonale

Procedura `vptridiag` are drept parametri o matrice simetrică tridiagonală a, dimensiunea lui a, şi eroarea cu care se aproximează valorile proprii ale lui a. Procedura întoarce un vector ce conține aproximații ale valorilor proprii ale lui a. Procedura `vptridiagk` are drept parametri o matrice simetrică tridiagonală a, dimensiunea lui a, un număr natural nenul k mai mic sau egal cu dimensiunea lui a, şi eroarea cu care se aproximează valorile proprii ale lui a. Procedura întoarce aproximația celei de a k-a valori proprii ale lui a (valorile proprii se presupun ordonate).

```
>vptridiag := proc(a, n, eps)
local vp, i, j, k, nv, a0, b0, c, norma, p0, p1, p2;
    vp := vector(n);
    norma := abs(a[1, 1]) + abs(a[1, 2]);
    for i from 2 to n - 1 do
        a0 := 0;
        for j from i - 1 to i + 1 do a0 := a0 +
abs(a[i, j]) od;
        if norma < a0 then norma := a0 fi
    od;
    if norma < abs(a[n, n - 1]) + abs(a[n, n]) then
        norma := abs(a[n, n - 1]) + abs(a[n, n])
    fi;
    c := -norma;
    b0 := norma;
    for k to n do
        a0 := c;
        b0 := norma;
        while eps < abs(b0 - a0) do
```

```
            c := evalf(1/2*a0 + 1/2*b0);
            p0 := evalf(1);
            p1 := evalf(a[1, 1] - c);
            nv := 0;
            for i from 2 to n do
                if p0*p1 < 0 then nv := nv + 1 fi;
                p2 := (a[i, i] - c)*p1 - a[i, i -
1]^2*p0;
                p0 := p1;
                p1 := p2
            od;
            if p0*p1 < 0 then nv := nv + 1 fi;
            if k <= nv then b0 := c else a0 := c fi
        od;
        vp[k] := c
    od;
    RETURN(evalm(vp))
end;


>vptridiagk := proc(a, n, k, eps)
local vp, i, j, nv, a0, b0, c, norma, p0, p1, p2;
    norma := abs(a[1, 1]) + abs(a[1, 2]);
    for i from 2 to n - 1 do
        a0 := 0;
        for j from i - 1 to i + 1 do a0 := a0 +
abs(a[i, j]) od;
        if norma < a0 then norma := a0 fi
    od;
    if norma < abs(a[n, n - 1]) + abs(a[n, n]) then
        norma := abs(a[n, n - 1]) + abs(a[n, n])
    fi;
    c := -norma;
    b0 := norma;
    a0 := c;
    b0 := norma;
    while eps < abs(b0 - a0) do
        c := evalf(1/2*a0 + 1/2*b0);
        p0 := evalf(1);
        p1 := evalf(a[1, 1] - c);
        nv := 0;
        for i from 2 to n do
            if p0*p1 < 0 then nv := nv + 1 fi;
```

```
            p2 := (a[i, i] - c)*p1 - a[i, i -
1]^2*p0;
            p0 := p1;
            p1 := p2
        od;
        if p0*p1 < 0 then nv := nv + 1 fi;
        if k <= nv then b0 := c else a0 := c fi
    od;
    vp := c;
    RETURN(vp)
end;
```

Exemple

```
> a:=matrix(3,3,[1,2,0,2,1,2,0,2,1]);
```

$$a := \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$$

```
> evalf(Eigenvals(a));
      [-1.828427124, 1.000000001, 3.828427124]
> vptridiag(a,3,0.00001);
      [-1.828432085, 1.000001001, 3.828422845]
> vptridiagk(a,3,2,0.000001);
                        .9999996428
> b:=matrix(3,3,[1,0,0,0,1,0,0,0,1]);
```

$$b := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
> vptridiag(b,3,0.0000001);
       [.9999999404, .9999999404, .9999999404]
> evalf(Eigenvals(b));
                                [1., 1., 1.]
> vptridiagk(b,3,3,0.000001);
                        .9999990464
> c:=matrix(3,3,[2,0,0,0,1,0,0,0,2]);
```

$$\begin{pmatrix} 2 & 0 & 0 \\ & & \end{pmatrix}$$

```
                c :=     0    1    0
                         0    0    2

> vptridiag(c,3,0.0000001);
      [1.000000060, 1.999999941, 1.999999941]
> evalf(Eigenvals(c));
                    [1., 2., 2.]
> d:=matrix(4,4,[1,2,0,0,2,1,2,0,0,2,2,3,0,0,3,1]);
```

$$d:=\begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & 1 & 2 & 0 \\ 0 & 2 & 2 & 3 \\ 0 & 0 & 3 & 1 \end{pmatrix}$$

```
> evalf(Eigenvals(d));

     [-2.416129024, -.5053965531, 2.631926621,
                    5.289598958]

> vptridiag(d,4,0.00001);
     [-2.416123393, -.5054040117, 2.631928108,
                    5.289597862]
> vptridiagk(d,4,2,0.000001);
                    -.5053962473
```

Procedură MAPLE pentru determinarea valorilor proprii ale unei matrice simetrice

Procedura `GHouseholder` are drept parametrii a matrice simetrică b, dimensiunea matricei, şi eroarea cu care se aproximează valorile proprii ale lui b. Procedura întoarce un vector ce conține valorile proprii ale lui b. Algoritmul utilizat este algoritmul Givens- Householder. În prima etapă se determină o matrice a simetrică tridiagonală similară cu b şi apoi se apelează procedura `vptridiag` pentru a. Este afişată şi matricea tridiagonală obținută.

```
>GHouseholder := proc(b, n, eps)
local a, i, j, k, w, q, suma, p;
    a := matrix(n, n);
    w := vector(n);
    q := vector(n);
    p := vector(n);
    for i to n do for j from i to n do
```

```
            a[i, j] := evalf(b[i, j]); a[j, i] :=
a[i, j]
        od
    od;
    for k to n - 2 do
        suma := 0;
        for i from k + 1 to n do
            suma := suma + a[i, k]^2; w[i] := a[i,
k]
        od;
        suma := suma^(1/2);
        if a[k + 1, k] < 0 then
            a[k, k + 1] := suma; w[k + 1] := w[k +
1] - suma
        else a[k, k + 1] := -suma; w[k + 1] := w[k +
1] + suma
        fi;
        a[k + 1, k] := a[k, k + 1];
        suma := 0;
        for i from k + 1 to n do suma := suma +
w[i]^2 od;
        if 0 < suma then
            suma := suma^(1/2);
            for i from k + 1 to n do w[i] :=
w[i]/suma od;
            for i from k + 1 to n do
                p[i] := 0;
                for j from k + 1 to n do
                    p[i] := p[i] + a[i, j]*w[j]
                od
            od;
            for i from k + 1 to n do
                suma := 0;
                for j from k + 1 to n do suma :=
suma + w[j]*p[j]
                od;
                q[i] := 2*p[i] - 2*w[i]*suma
            od;
            for i from k + 1 to n do for j from i to
n do
                    a[i, j] := a[i, j] - w[i]*q[j] -
q[i]*w[j];
                    a[j, i] := a[i, j]
```

```
                od
            od
        fi
    od;
    for i to n do
        for j from i + 2 to n do a[i, j] := 0; a[j,
i] := 0 od
    od;
    print(a);
    RETURN(vptridiag(a, n, eps))
end
```

Exemple

```
> a1:=matrix(3,3,[1,2,3,2,4,5,3,5,6]);
```

$$a1 := \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix}$$

➢ `GHouseholder(a1,3,0.00001);`

$$\begin{pmatrix} 1. & -3.605551275 & 0 \\ -3.605551275 & 9.999999998 & .999999997 \\ 0 & .999999997 & -.4\ 10^{-8} \end{pmatrix}$$

```
      [-.5157259363, .1709119936, 11.34481884]

> evalf(Eigenvals(a1));

      [-.5157294711, .1709151916, 11.34481429]

>a2:=matrix(4,4,[1,2,3,7,2,4,5,8,3,5,6,9,7,8,9,10]);
```

$$a2 := \begin{pmatrix} 1 & 2 & 3 & 7 \\ 2 & 4 & 5 & 8 \\ 3 & 5 & 6 & 9 \\ 7 & 8 & 9 & 10 \end{pmatrix}$$

```
> GHouseholder(a2,4,0.000001);
```

$$\begin{pmatrix} 1. & -7.874007874 & 0 & 0 \\ -7.874007874 & 19.70967742 & 7.190690875 & 0 \end{pmatrix}$$

```
 0 ,          7.190690875 ,.3351509578 ,.0332378657
 0 ,       0 ,          .0332378657,   -.0448283894
```

```
     [-4.101021296, -.04548288337, .6567460981,
                   24.48975801]
```

```
> evalf(Eigenvals(a2));
```

```
     [-4.101020873, -.04548283154, .6567457633,
                   24.48975796]
```

```
> a3:=matrix(3,3,[1,0,-1,0,2,1,-1,1,0]);
```

$$a3 := \begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

```
> GHouseholder(a3,3,0.00001);
```

$$\begin{pmatrix} 1. & -1.000000000 & -1. \\ -1.000000000 & -.2\ 10^{-8} & 1.000000000 \\ -1. & 1.000000000 & 2.000000002 \end{pmatrix}$$

```
      [-.8793811808, 1.347292521, 2.532093088]
```

```
> evalf(Eigenvals(a3));
      [-.879385242, 1.347296355, 2.532088886]
a4:=matrix(4,4,[1,2,0,0,2,1,2,0,0,2,2,3,0,0,3,1]);
```

$$a4 := \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & 1 & 2 & 0 \\ 0 & 2 & 2 & 3 \\ 0 & 0 & 3 & 1 \end{pmatrix}$$

```
> evalf(Eigenvals(a4));
```

```
     [-2.416129024, -.5053965531, 2.631926621,
                   5.289598958]
```

```
> GHouseholder(a4,4,0.00001);
```

```
⎛ 1.          2.000000000    0          0 ⎞
⎜ 2.000000000     1.     2.000000000   0 ⎟
⎜ 0     2.000000000     2.     3.000000000⎟
⎝ 0     0               3.000000000    1 ⎠
```

```
      [-2.416123393, -.5054040117, 2.631928108,
                    5.289597862]
```

```
>a5:=matrix(5,5,[1,2,3,7,11,2,4,5,8,12,3,5,6,9,11,7,
8,9,10,10,11,12,11,10,1]);
```

$$a5 := \begin{pmatrix} 1 & 2 & 3 & 7 & 11 \\ 2 & 4 & 5 & 8 & 12 \\ 3 & 5 & 6 & 9 & 11 \\ 7 & 8 & 9 & 10 & 10 \\ 11 & 12 & 11 & 10 & 1 \end{pmatrix}$$

```
> GHouseholder(a5,5,0.0001);
```

```
⎛  1. ,      -13.52774926 ,      0 ,      0 ,     0 ⎞
⎜-13.52774926 ,22.60655740 ,18.80510804 ,0 , 0      ⎟
⎜0 ,18.80510804 ,-1.035783268 ,-1.207071098 ,0      ⎟
⎜0 ,0 ,-1.207071098 ,-.7873820046 ,-.1131583302     ⎟
⎝0 ,   0 ,   0 ,   -.1131583302 ,   .2166078862    ⎠
```

```
      [-14.81747731, -1.063825571, .2194289136,
              .6642020530, 36.99759394]

> evalf(Eigenvals(a5));

      [-14.81743076, -1.063820286, .2194285474,
              .6642185938, 36.99760395]
```

III.4. Calculul vectorilor şi valorilor proprii pentru matrice oarecare. Metoda puterii. Algoritmul QR

Cea mai simplă metodă de calcul a unui vector propriu şi a valorii proprii asociate este ***metoda puterii***. Se presupune că multiplicitatea algebrică şi multiplicitatea geometrică a valorilor proprii ale matricei A coincid, sau echivalent că matricea n – dimensională A are un sistem de n vectori liniar independenţi. Se pleacă de la un vector $x_0 \in \mathbf{R}^n$ nenul şi se construieşte şirul $(x_k)_k$:

$$x_k = Ax_{k-1},\ k \geq 1.$$

Dacă există o unică valoarea proprie dominantă a lui A, λ_1, adică

$$|\lambda_1| > |\lambda_i|,\ i = 2,3,\ldots,n.$$

atunci şirul $(x_k)_k$ converge la un vector propriu asociat lui λ_1. Pentru a înţelege de ce şirul $(x_k)_k$ converge să considerăm n vectori proprii liniar independenţi $v_1, v_2, \ldots, v_n$ şi să exprimăm pe x_0 în baza din $\mathbf{R}^n$ determinată de aceşti vectori:

$$x_0 = c_1v_1 + c_2v_2 + \ldots + c_nv_n$$

Atunci

$$\begin{aligned} x_k &= Ax_{k-1} = A^2x_{k-2} = \ldots = A^kx_0 \\ &= c_1A^kv_1 + c_2A^kv_2 + \ldots + c_nA^kv_n \\ &= c_1\lambda_1^k v_1 + c_2\lambda_2^k v_2 + \ldots + c_n\lambda_n^k v_n \\ &= \lambda_1^k (c_1v_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \ldots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k v_n). \end{aligned}$$

Deoarece $\left|\frac{\lambda_i}{\lambda_1}\right| < 1$ pentru orice i =2,3,…, n, $\lim_{k\to\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$. Deci pentru k suficient de mare rămâne doar termenul asociat lui v_1. În cazul unei matrice oarecare (care nu îndeplineşte condiţiile precedente), metoda puterii directe poate eşua din mai multe motive:

1) vectorul de plecare x_0 are componenta $c_1 = 0$ – în practică aceasta nu este o problemă deoarece eroarea de rotunjire introduce de obicei o componentă nenulă.
2) există mai multe valori proprii maxime în modul – caz în care şirul converge la o combinaţie liniară de vectori proprii (asociaţi valorilor proprii maime în modul)
3) în cazul unei matrice şi unui vector de plecare cu componente reale, şirul nu converge la un vector cu componente complexe (nereale)

Datorită iteraţiilor componentele vectorului x_k pot creşte/scădea în afara limitelor de reprezentare (overflow/underflow). Pentru a evita aceasta la fiecare iteraţie vectorul ce aproximează vectorul propriu se normalizează astfel încât norma lui infinit să fie 1, adică cea mai mare componentă în

modul să aibă modul egal cu 1. Astfel ***metoda puterii directe*** presupune construcția șirurilor:

$y_k = Ax_k$, $k \geq 0$

$x_{k+1} = y_k / \|y_k\|_\infty$, $k \geq 0$.

cu x_0 dat. . $(\|y_k\|_\infty)_k$ converge la $|\lambda_1|$ și $(x_k)_k$ converge la $v_1/\|v_1\|_\infty$.

Să presupunem că x_{k+1} reprezintă o aproximație suficient de bună pentru vectorul propriu v_1 (presupumem că se dă $\varepsilon > 0$, iar k+1 este numărul iterației pentru care $\left| \|y_{k+1}\|_\infty - \|y_k\|_\infty \right| < \varepsilon$). Pentru determinarea unei aproximații a valorii proprii λ_1 putem observa că

$$\frac{x_{k+1}}{x_k} \approx \lambda_1 \text{ adică } \frac{(x_{k+1})_i}{(x_k)_i} \approx \lambda_1 \text{ pentru orice } i = 1,2,\ldots, n$$

și putem aproxima λ_1 prin media aritmetică a componentelor lui x_{k+1} împărțită la media aritmetică a componentelor lui x_k. O variantă mai bună de aproximare folosește câtul Rayleigh. Se pleacă de la observația că dacă x este un vector propriu al matricei A, atunci raportul $\frac{\langle Ax, x\rangle}{\langle x, x\rangle}$ reprezintă valoarea proprie asociată lui x. Astfel dacă x_k reprezintă o aproximație suficient de bună pentru vectorul propriu v_1 atunci

$$\frac{\langle Ax_k, x_k\rangle}{\langle x_k, x_k\rangle} \approx \lambda_1$$

reprezintă o aproximație pentru λ_1.

Metoda puterii directe cu deplasarea originii presupune aplicarea metodei puterii directe unei matrice de forma A – qI, cu q convenabil ales. Se pleacă de la observația că matricele A și A – qI au aceiași vectori proprii iar valorile proprii ale lui A se obțin din valorile proprii ale lui A – qI la care se adună q. Dacă

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$$

sunt valorile proprii ale lui A și alegem q astfel încât

$$\left|\frac{\lambda_2 - q}{\lambda_1 - q}\right| < \left|\frac{\lambda_2}{\lambda_1}\right|$$

atunci

$$|\lambda_1 - q| > |\lambda_2 - q| \geq \ldots$$

sunt valorile proprii ale matricei A –qI, iar convergența este accelerată. Dacă se alege $q = (\lambda_n + \lambda_2)/2$ se obține rata maximă de convergență a șirului ce converge la vectorul propriu asociat lui $\lambda_1 - q$, iar dacă se alege $q = (\lambda_{n-1} +$

$\lambda_1)/2$ se obține rata maximă de convergență a şirului ce converge la vectorul propriu asociat lui $\lambda_n - q$.

Metoda puterii inverse presupune aplicarea metodei puterii directe inversei matricei A. Plecând de la observația că valorile proprii ale inversei matricei A sunt inversele valorilor proprii ale lui A, rezultă că şirul construit va converge la un vector propriu asociat valorii proprii cu modulul cel mai mic. Astfel metoda puterii inverse presupune construcția şirurilor:

$Ay_k = x_{k-1}$, $k \geq 1$

$x_k = y_{k/} \|y_k\|_\infty$, $k \geq 0$.

cu x_0 dat. $(\|y_k\|_\infty)_k$ converge la $|\lambda_n|$ şi $(x_k)_k$ converge la $v_n/\|v_n\|_\infty$, unde λ_n este valoarea proprie cu modulul cel mai mic, iar v_n este un vector propriu asociat lui λ_n. Aplicând această metodă matricei A- qI se obține valoarea proprie a lui A cea mai apropiată de q. De aici rezultă că pentru o alegere atentă a lui q se poate determina orice valoare proprie a lui A folosind procesul iterativ:

$(A-qI)y_k = x_{k-1}$, $k \geq 1$

$x_k = y_{k/} \|y_k\|_\infty$, $k \geq 0$.

O altă metodă de determinare a tuturor valorilor asociate unei matrice este ***deflația***. Odată determinată valoarea proprie λ_1 şi un vector propriu x_1, se trece la determinarea valorilor proprii $\lambda_2, \lambda_3, \ldots, \lambda_n$ prin înlăturarea lui λ_1. Se poate proceda în mai multe moduri. Fie H o matrice Householder cu proprietatea că $Hx_1 = \alpha e_1$, unde e_1 este prima coloană din matricea unitate. Atunci

$$HAH^{-1} = \begin{pmatrix} \lambda_1 & b^t \\ O & B \end{pmatrix}$$

unde B este o matrice de dimensiune n-1 ce având valorile proprii $\lambda_2, \lambda_3, ..\lambda_n$. Se aplică metoda puterii pentru determinarea următoarei valori proprii λ_2 şi a vectorului propriu corespunzător, y_2. Atunci

$$x_2 = H^{-1}\begin{pmatrix} \alpha \\ y_2 \end{pmatrix},$$

unde $\alpha = \dfrac{b^t y_2}{\lambda_2 - \lambda_1}$, este vector propriu al lui A asociat lui λ_2.

Alternativ se poate alege u_1 astfel încât ${}^tu_1x_1 = \lambda_1$. Atunci $A - x_1{}^tu_1$are valorile proprii 0, λ_2, …, λ_n. Se poate aplica mai departe metoda puterii directe matricei $A - x_1{}^tu_1$ pentru determinarea următoarei valori.

Metoda QR este o metodă iterativă de construcție a unui şir $(A_k)_k$ de matrice similare , şir convergent la o matrice superior triunghiulară. Această metodă foloseşte factorizarea QR a matricelor A_k.

Factorizarea QR a unei matrice A presupune reprezentarea A = QR, unde Q este o matrice ortogonală, iar R este o matrice superior triunghiulară. Procedura de factorizare QR a matricei n dimensionale A se realizează în n-1 etape. Se pleacă cu $A_1 = A$, iar dacă A_k este matricea obținută după k-1 etape atunci

$$A_{k+1} = H_kA_k,$$

unde H_k este matricea Householder ce lasă invariante liniile şi coloanele 1,2,…, k-1 şi anulează elementele subdiagonle din coloane k. Matricea A_n va fi superior triunghiulară. Deci $A = QR$, unde $Q = H_1H_2…H_n$, iar $R = A_n$.

Metoda QR este decrisă de formulele

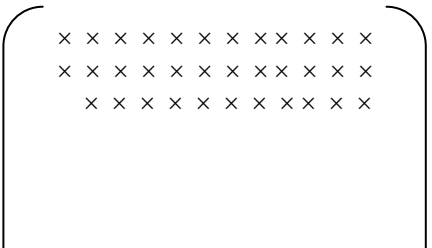
$$\begin{cases} A_1 = A \\ A_k = Q_kR_k,\ k\geq 1 \text{(factorizare QR a matricei } A_k) \\ A_{k+1} = R_kQ_k,\ k\geq 1 \end{cases}$$

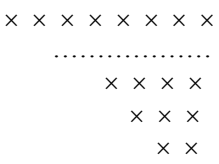
Matricele A_k sunt similare cu A, deoarece se observă că

$$A_k = R_{k-1}Q_{k-1} = Q^{-1}{}_{k-1}A_{k-1}\,Q_{k-1},\ k\geq 2.$$

Dacă A este o matrice inversabilă ale cărei valori proprii sunt distincte în modul atunci şirul de matrice $(A_k)_k$ converge la o matrice superior triunghiulară ce are pe diagonala principală valorile proprii ale lui A. Produsul matricelor $Q_1Q_2…Q_k$ converge la o matrice ale cărei coloane sunt vectori proprii pentru A.

Pentru micşorarea numărului de operații se recomandă aducerea matricei A la forma Hessenberg superioară (formă aproape superior triunghiulră):





Pentru aducerea matricei A la forma Hessenberg superioară se poate plica metoda Householder, descrisă în prima etapa a algoritmului de calcul a valorilor proprii asociate unei matrice simetrice (etapa în care se transforma A într-o matrice tridiagonală). Astfel se recomandă implementarea algoritmului QR în două etape:

matrice simetrică → matrice tridiagonală → matrice diagonală

matrice oarecare → matrice Hessenberg superioară →matrice triunghiulară

Proceduri MAPLE pentru calculul vectorilor şi valorilor proprii.

Procedura `puteredirecta` are drept parametri o matrice a, dimensiunea lui a, un vector n –dimensional x00, şi eroarea cu care se aproximează valoarea proprie maximă în modul. Procedura afişează aproximaţii ale valorii proprii maxime în modul, şi a vectorului propriu corespunzător. Acestea sunt determinate prin aplicarea metodei puterii directe. Pentru fiecare iteraţie k sunt afişaţi x_k, $y_k = ax_k$.

Procedura `putereinversa` este similară. Diferenţa constă în faptul că se aplică metoda puterii inverse.

Procedura `pinv` are drept parametri o matrice b, dimensiunea lui b, un vector n –dimensional x00, eroarea cu care se aproximează valoarea proprie cea mai apropiată de q, ultimul parametru. Procedura întoarce aproximaţii ale valorii proprii celei mai apropiate de q, şi a vectorului propriu corespunzător. Acestea sunt determinate prin aplicarea metodei puterii inverse cu deplasarea originii.

```
>puteredirecta := proc(a, n, x00, eps)
local i, j, x0, x1, norma, n0, n1, l1, l2, s;
    x0 := vector(n);
    for i to n do x0[i] := evalf(x00[i]) od;
    x1 := vector(n);
    n0 := eps + 1;
    n1 := 0;
    while eps <= abs(n0 - n1) do
        n0 := n1;
        for i to n do
```

```
            x1[i] := 0;
            for j to n do x1[i] := x1[i] + a[i,
j]*x0[j] od
        od;
        print(x0, x1);
        norma := x1[1];
        for i from 2 to n do
            if norma < abs(x1[i]) then norma :=
abs(x1[i]) fi
        od;
        for i to n do x1[i] := x1[i]/norma od;
        n1 := norma;
        for j to n do x0[j] := x1[j] od
    od;
    l1 := 0;
    for i to n do for j to n do l1 := l1 + a[i,
j]*x1[j] od od;
    s := 0;
    for i to n do s := s + x1[i] od;
    l1 := l1/s;
    l2 := 0;
    for i to n do for j to n do l2 := l2 + a[i,
j]*x1[j]*x1[i] od
    od;
    s := 0;
    for i to n do s := s + x1[i]^2 od;
    l2 := l2/s;
    print(`aprox. val prop. maxima`, l1);
    print(`aprox.Rayleigh val prop. maxima`, l2);
    print(`vector propriu`, x1)
end;

>putereinversa := proc(a, n, x00, eps)
local i, j, x0, x1, norma, n0, n1, l1, l2, s;
    x0 := vector(n);
    for i to n do x0[i] := evalf(x00[i]) od;
    x1 := vector(n);
    n0 := eps + 1;
    n1 := 0;
    while eps <= abs(n0 - n1) do
        n0 := n1;
        x1 := linsolve(a, x0);
        print(x0, x1);
```

```
        norma := x1[1];
        for i from 2 to n do
            if norma < abs(x1[i]) then norma :=
abs(x1[i]) fi
        od;
        for i to n do x1[i] := x1[i]/norma od;
        n1 := norma;
        for j to n do x0[j] := x1[j] od
    od;
    l1 := 0;
    for i to n do for j to n do l1 := l1 + a[i,
j]*x1[j] od od;
    s := 0;
    for i to n do s := s + x1[i] od;
    l1 := l1/s;
    l2 := 0;
    for i to n do for j to n do l2 := l2 + a[i,
j]*x1[j]*x1[i] od
    od;
    s := 0;
    for i to n do s := s + x1[i]^2 od;
    l2 := l2/s;
    print(`aprox. val prop. minima`, l1);
    print(`aprox.Rayleigh val prop. minima`, l2);
    print(`vector propriu`, x1)
end;

>pinv := proc(b, n, x00, eps, q)
local i, j, x0, x1, norma, n0, n1, l2, s, a;
    x0 := vector(n);
    a := matrix(n, n);
    for i to n do x0[i] := evalf(x00[i]) od;
    x1 := vector(n);
    n0 := eps + 1;
    n1 := 0;
    for i to n do for j to n do a[i, j] := b[i, j]
od od;
    for i to n do a[i, i] := a[i, i] - q od;
    while eps <= abs(n0 - n1) do
        n0 := n1;
        for i to n do
            x1[i] := 0;
```

```
            for j to n do x1[i] := x1[i] + a[i,
j]*x0[j] od
        od;
        norma := x1[1];
        for i from 2 to n do
            if norma < abs(x1[i]) then norma :=
abs(x1[i]) fi
        od;
        for i to n do x1[i] := x1[i]/norma od;
        n1 := norma;
        for j to n do x0[j] := x1[j] od
    od;
    l2 := 0;
    for i to n do for j to n do l2 := l2 + a[i,
j]*x1[j]*x1[i] od
    od;
    s := 0;
    for i to n do s := s + x1[i]^2 od;
    l2 := l2/s;
    RETURN([l2 + q, evalm(x1)])

end;
```

Exemple

```
> a1:=matrix(2,2,[1.5,0.5,0.5,1.5]);
```

$$a1 := \begin{bmatrix} 1.5 & 0.5 \\ .5 & 1.5 \end{bmatrix}$$

```
> a2:=matrix(3,3,[1,3,-1,1,-1,1,0,1,-2]);
```

$$a2 := \begin{bmatrix} 1 & 3 & -1 \\ 1 & -1 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

```
> x01:=vector(2,[0,1]);
                    x01 := [0, 1]
> x02:=vector(2,[0,0.1]);
                   x02 := [0, .1]
> x03:=vector(3,[1,0,0]);
                  x03 := [1, 0, 0]
> eigenvects(a1);
  [1.000000000, 1, {[.7071067812, -.7071067812]}],
```

```
    [2.0, 1, {[-.7071067812, -.7071067812]}]

> eigenvects(a2);
[-3, 1, {[1, -1, 1]}], [2, 1, {[11, 4, 1]}], [-1, 1,
                    {[-1, 1, 1]}]
> puteredirecta(a1,2,x01,0.001);

                  [0, 1.], [.5, 1.5]
      [.3333333333, 1.000000000], [1.000000000,
                    1.666666667]
      [.5999999999, 1.000000000], [1.400000000,
                    1.800000000]
      [.7777777778, 1.000000000], [1.666666667,
                    1.888888889]
      [.8823529413, 1.000000000], [1.823529412,
                    1.941176471]
      [.9393939393, 1.000000000], [1.909090909,
                    1.969696970]
      [.9692307690, 1.000000000], [1.953846154,
                    1.984615385]
      [.9844961239, 1.000000000], [1.976744186,
                    1.992248062]
      [.9922178988, 1.000000000], [1.988326848,
                    1.996108949]
      [.9961013646, 1.000000000], [1.994152047,
                    1.998050682]
      [.9980487807, 1.000000000], [1.997073171,
                    1.999024390]
        aprox. val prop. maxima, 2.000000001
    aprox.Rayleigh val prop. maxima, 1.999999761
     vector propriu, [.9990239144, 1.000000000]

> puteredirecta(a1,2,x01,0.000001);

                  [0, 1.], [.5, 1.5]
      [.3333333333, 1.000000000], [1.000000000,
                    1.666666667]
      [.5999999999, 1.000000000], [1.400000000,
                    1.800000000]
      [.7777777778, 1.000000000], [1.666666667,
                    1.888888889]
      [.8823529413, 1.000000000], [1.823529412,
                    1.941176471]
```

```
      [.9393939393, 1.000000000], [1.909090909,
                    1.969696970]
      [.9692307690, 1.000000000], [1.953846154,
                    1.984615385]
      [.9844961239, 1.000000000], [1.976744186,
                    1.992248062]
      [.9922178988, 1.000000000], [1.988326848,
                    1.996108949]
      [.9961013646, 1.000000000], [1.994152047,
                    1.998050682]
      [.9980487807, 1.000000000], [1.997073171,
                    1.999024390]
      [.9990239144, 1.000000000], [1.998535872,
                    1.999511957]
      [.9995118384, 1.000000000], [1.999267758,
                    1.999755919]
      [.9997558897, 1.000000000], [1.999633835,
                    1.999877945]
      [.9998779376, 1.000000000], [1.999816906,
                    1.999938969]
      [.9999389666, 1.000000000], [1.999908450,
                    1.999969483]
      [.9999694830, 1.000000000], [1.999954225,
                    1.999984742]
      [.9999847414, 1.000000000], [1.999977112,
                    1.999992371]
      [.9999923705, 1.000000000], [1.999988556,
                    1.999996185]
      [.9999961855, 1.000000000], [1.999994278,
                    1.999998093]
      [.9999980925, 1.000000000], [1.999997139,
                    1.999999046]
         aprox. val prop. maxima, 1.999999999
     aprox.Rayleigh val prop. maxima, 2.000000000
      vector propriu, [.9999990465, 1.000000000]

> putereinversa(a1,2,x01,0.001);

         [0, 1.], [-.2500000001, .7500000002]
     [-.3333333334, 1.000000000], [-.5000000002,
                    .8333333335]
     [-.6000000001, 1.000000000], [-.7000000003,
                    .9000000002]
```

```
    [-.7777777779, 1.000000000], [-.8333333335,
                    .9444444446]
    [-.8823529412, 1.000000000], [-.9117647061,
                    .9705882355]
    [-.9393939394, 1.000000000], [-.9545454550,
                    .9848484851]
    [-.9692307694, 1.000000000], [-.9769230775,
                    .9923076926]
    [-.9844961243, 1.000000000], [-.9883720938,
                    .9961240314]
    [-.9922178992, 1.000000000], [-.9941634250,
                    .9980544751]
    [-.9961013650, 1.000000000], [-.9970760238,
                    .9990253414]
       aprox. val prop. minima, 2.000000103
   aprox.Rayleigh val prop. minima, 1.000000953
    vector propriu, [-.9980487806, 1.000000000]
> putereinversa(a1,2,x01,0.000001);
       [0, 1.], [-.2500000001, .7500000002]
    [-.3333333334, 1.000000000], [-.5000000002,
                    .8333333335]
    [-.6000000001, 1.000000000], [-.7000000002,
                    .9000000002]
    [-.7777777778, 1.000000000], [-.8333333335,
                    .9444444445]
    [-.8823529413, 1.000000000], [-.9117647065,
                    .9705882356]
    [-.9393939397, 1.000000000], [-.9545454550,
                    .9848484851]
    [-.9692307694, 1.000000000], [-.9769230772,
                    .9923076925]
    [-.9844961241, 1.000000000], [-.9883720938,
                    .9961240314]
    [-.9922178992, 1.000000000], [-.9941634247,
                    .9980544753]
    [-.9961013645, 1.000000000], [-.9970760236,
                    .9990253413]
    [-.9980487805, 1.000000000], [-.9985365858,
                    .9995121954]
    [-.9990239143, 1.000000000], [-.9992679363,
                    .9997559789]
    [-.9995118383, 1.000000000], [-.9996338793,
                    .9998779599]
```

```
     [-.9997558896, 1.000000000], [-.9998169175,
                      .9999389726]
     [-.9998779375, 1.000000000], [-.9999084535,
                      .9999694846]
     [-.9999389670, 1.000000000], [-.9999542253,
                      .9999847419]
     [-.9999694829, 1.000000000], [-.9999771123,
                      .9999923709]
     [-.9999847413, 1.000000000], [-.9999885565,
                      .9999961856]
     [-.9999923709, 1.000000000], [-.9999942783,
                      .9999980929]
     [-.9999961854, 1.000000000], [-.9999971395,
                      .9999990466]
        aprox. val prop. minima, 1.999895129
    aprox.Rayleigh val prop. minima, .9999999995
     vector propriu, [-.9999980929, 1.000000000]

> puteredirecta(a2,3,x03,0.01);

                [1., 0, 0], [1., 1., 0]
   [1.000000000, 1.000000000, 0], [4.000000000, 0,
                      1.000000000]
            [1.000000000, 0, .2500000000],
             [.7500000000, 1.250000000, -.5000000000]
      [.6000000000, 1.000000000, -.4000000000],
             [4.000000000, -.8000000000, 1.800000000]
      [1.000000000, -.2000000000, .4500000000],
            [-.0500000000, 1.650000000, -1.100000000]
     [-.03030303030, 1.000000000, -.6666666667],
             [3.636363637, -1.696969697, 2.333333333]
      [1.000000000, -.4666666666, .6416666665],
            [-1.041666667, 2.108333334, -1.750000000]
     [-.4940711462, 1.000000000, -.8300395254],
             [3.335968379, -2.324110671, 2.660079051]
      [1.000000000, -.6966824643, .7973933649],
            [-1.887440758, 2.494075829, -2.291469194]
     [-.7567695962, 1.000000000, -.9187648456],
             [3.161995250, -2.675534442, 2.837529691]
      [1.000000000, -.8461538461, .8973858171],
            [-2.435847355, 2.743539663, -2.640925480]
     [-.8878484200, 1.000000000, -.9625978861],
             [3.074749466, -2.850446306, 2.925195772]
```

```
      [1.000000000, -.9270499394, .9513606895],
            [-2.732510508, 2.878410629, -2.829771318]
      [-.9493122630, 1.000000000, -.9831020249],
             [3.033789762, -2.932414288, 2.966204050]
      [1.000000000, -.9665845421, .9777223482],
            [-2.877475974, 2.944306890, -2.922029238]
      [-.9773016474, 1.000000000, -.9924336515],
             [3.015132005, -2.969735299, 2.984867303]
      [1.000000000, -.9849437086, .9899623957],
            [-2.944793522, 2.974906105, -2.964868500]
      [-.9898778039, 1.000000000, -.9966259086],
             [3.006748105, -2.986503713, 2.993251817]
      [1.000000000, -.9932670143, .9955113340],
            [-2.975312377, 2.988778348, -2.984289682]
      [-.9954944899, 1.000000000, -.9984981603],
             [3.003003670, -2.993992650, 2.996996321]
      [1.000000000, -.9969993310, .9979995532],
            [-2.988997546, 2.994998884, -2.992998437]
        aprox. val prop. maxima, -3.010715489
    aprox.Rayleigh val prop. maxima, -3.001779254
    vector propriu, [-.9979962136, 1.000000000, -
                    .9993320709]

> putereinversa(a2,3,x03,0.01);

 [1., 0, 0], [.1666666667, .3333333333, .1666666667]
       [.5000000002, 1.000000000, .5000000002],
            [1.083333333, -.3333333332, -.4166666667]
      [1.000000000, -.3076923077, -.3846153848],
             [-.2179487181, .5641025640, .4743589744]
      [-.3863636367, 1.000000000, .8409090911],
            [1.049242425, -.7424242429, -.7916666671]
      [1.000000000, -.7075812274, -.7545126352],
             [-.6744885678, .8206979543, .7876052948]
      [-.8218475071, 1.000000000, .9596774193],
            [1.016251222, -.9271749756, -.9434261975]
      [1.000000000, -.9123482024, -.9283395455],
             [-.9030700166, .9468959158, .9376177307]
      [-.9537162443, 1.000000000, .9902014731],
            [1.004447784, -.9813059060, -.9857536896]
      [1.000000000, -.9769605963, -.9813886847],
             [-.9745967255, .9861164269, .9837525559]
      [-.9883181123, 1.000000000, .9976028480],
```

```
           [1.001147931, -.9953069868, -.9964549174]
     [1.000000000, -.9941657531, -.9953123675],
            [-.9935755835, .9964927069, .9959025373]
        aprox. val prop. minima, -.9906474196
    aprox.Rayleigh val prop. minima, -1.001169780
     vector propriu, [-.9970726094, 1.000000000,
                     .9994077532]

> pinv(a1,2,x01,0.0001,8);
     [1.000000199, [-.9991015469, 1.000000000]]
> pinv(a2,3,x03,0.00001,7);

     [-2.999989846, [1.000000000, -1.000000000,
                    .9999695525]]
> pinv(a2,3,x03,0.00000001,0);
    [-3.000000003, [-.9999999977, 1.000000000, -
                    .9999999990]]
```