Autovalori

30 giugno 2007

1 Calcolo degli autovalori di una matrice

Il problema del calcolo degli autovalori di una matrice quadrata A di ordine n consiste nel trovare gli n numeri (possibilmente complessi) λ tali che

$$Ax = \lambda x,\ x \neq 0 \tag{1}$$

Si osservi che a seconda delle esigenze talvolta è richiesto solamente il calcolo di alcuni autovalori (ad esempio quelli di massimo modulo, per determinare lo spettro della matrice) mentre in altri casi si vogliono determinare tutti gli n autovalori.

Per semplicità mostreremo solo due metodi classici, uno per ognuna di queste classi, quello delle potenze e il metodo QR, rimandando per completezza alla monografia di Saad o a manuali di algebra lineare [2].

2 Il metodo delle potenze

Il metodo delle potenze è stato suggerito nel 1913 da Muntz ed è particolarmente indicato per il calcolo dell'autovalore di massimo modulo di una matrice.

Sia A una matrice quadrata di ordine n con n autovettori $x_1, \ldots, x_n$ linearmente indipendenti e autovalori $\lambda_1, \ldots, \lambda_n$ tali che

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|. \tag{2}$$

A tal proposito ricordiamo (cf. [3], p. 951) che

1. una matrice A ́diagonalizzabile se e solo se possiede n autovalori linearmente indipendenti;

2. Se tutti gli autovalori di A sono distinti la matrice è diagonalizzabile;

3. Una matrice simmetrica (hermitiana) è diagonalizzabile.

Il metodo delle potenze è definito come segue. Sia $t_0 \in \mathscr{R}^n$ definito da

$$t_0 = \sum_{i=1}^{n} \alpha_i x_i,\ \alpha_1 \neq 0,$$

e si generi la successione

$$y_0 = t_0 \tag{3}$$
$$y_k = Ay_{k-1},\ k = 1,2,\ldots \tag{4}$$

Si puó dimostrare che nelle ipotesi introdotte la successione $\{y_k\}$ converge ad un vettore parallelo a x_1 e che il coefficiente di Rayleigh (relativo al prodotto scalare euclideo)

$$\rho(y_k, A) := \frac{(y_k, Ay_k)}{(y_k, y_k)} \tag{5}$$

converge a λ_1.
Il metodo converge anche nel caso in cui

$$\lambda_1 = \ldots = \lambda_r$$

per $r > 1$, tuttavia non è da applicarsi quando l'autovalore di modulo massimo non è unico.
In virtú di alcune possibili problemi di underflow e underflow si preferisce normalizzare passo passo il vettore y_k definito in (3). Conseguentemente l'algoritmo del metodo delle potenze risulta

$$u_k = At_{k-1} \tag{6}$$
$$t_k = \frac{u_k}{\beta_k},\ \beta_k = \|u_k\|_2 \tag{7}$$
$$l_k = \rho(t_k, A) \tag{8}$$

dove $\rho(t_k, A)$ è il coefficiente di Rayleigh definito in (5).
Definiamo ora una versione di base `powerm` del metodo delle potenze

```
function [lambda1, x1, niter, err]=powerm(A,z0,toll,nmax)

% INPUT:
% A    : MATRICE DI CUI VOGLIAMO CALCOLARE L'AUTOVALORE DI MASSIMO MODULO.
% z0   : VETTORE INIZIALE (NON NULLO).
% toll: TOLLERANZA.
% nmax: NUMERO MASSIMO DI ITERAZIONI.
%
% OUTPUT:
% lambda1 : VETTORE DELLE APPROSSIMAZIONI DELL'AUTOVALORE DI MASSIMO
MODULO.
% x1       : AUTOVETTORE RELATIVO ALL'AUTOVALORE DI MASSIMO MODULO.
% niter    : NUMERO DI ITERAZIONI.
% err      : VETTORE DEI RESIDUI PESATI RELATIVI A "lambda1".
%
% TRATTO DA QUARTERONI-SALERI, "MATEMATICA NUMERICA", p. 184.
%
```

```
q=z0/norm(z0); q2=q; err=[]; lambda1=[];
res=toll+1; niter=0; z=A*q;
while (res >= toll & niter <= nmax)
    q=z/norm(z); z=A*q; lam=q'*z; x1=q;
    z2=q2'*A; q2=z2/norm(z2); q2=q2'; y1=q2; costheta=abs(y1'*x1);
    if (costheta > 5e-2)
        niter=niter+1; res=norm(z-lam*q)/costheta;
        err=[err; res]; lambda1=[lambda1; lam];
    else
        disp('\n \t [ATTENZIONE]: AUTOVALORE MULTIPLO'); break;
    end
end
```

Una breve analisi:

1. il vettore iniziale `z0` e' normalizzato ed in `err`, `lambda1` vengono memorizzati rispettivamente i valori dell'errore compiuto e dell'autovalore di massimo modulo $\lambda_{\max}$;

2. l'assegnazione `res=toll+1;` forza l'algoritmo ad entrare nel ciclo `while`, mentre `z=A*q;` è una quantità da utilizzarsi per il calcolo dell'autovalore $\lambda_{\max}$;

3. nel ciclo `while`, `q` è un'approssimazione di un autoversore relativo a $\lambda_{\max}$, mentre `lam` di $\lambda_{\max}$;

4. il ciclo si interrompe se un numero massimo di iterazioni `niter` è raggiunto oppure
$$\frac{||Aq^k - \lambda^k||_2}{|\cos(\theta_{\lambda_k})|} < \text{tol}$$
dove θ_{λ_k} è l'angolo formato tra (un'approssimazione del)l'autovalore destro `x1` e sinistro `y1` associati a `lam` (cf. [7, p.180]) oppure
$$|\cos(\theta_{\lambda_k})| < 5\,10^{-2}.$$

3 Il metodo delle potenze inverse

Una variante particolarmente interessante del metodo delle potenze è stata scoperta da Wielandt nel 1944 ed e' particolarmente utile nel caso

$$|\lambda_1| \geq |\lambda_2| \geq \ldots > |\lambda_n| > 0. \tag{9}$$

e si desideri calcolare il piú piccolo autovalore in modulo, cioè λ_n, applicando il metodo delle potenze ad A^{-1}. Si ottiene cosí la successione $\{t_k\}$ definita da

$$Au_k = t_{k-1} \tag{10}$$
$$t_k = \frac{u_k}{\beta_k},\ \beta_k = \|u_k\|_2 \tag{11}$$

e convergente ad un vettore parallelo a x_n. La successione di coefficienti di Rayleigh

$$\rho(t_k, A^{-1}) := \frac{(t_k, A^{-1}t_k)}{(t_k, t_k)} = \frac{(t_k, u_{k+1})}{(t_k, t_k)} \to 1/\lambda_n. \tag{12}$$

da cui è immediato calcolare λ_n.

Vediamo in dettaglio questo punto. Se $\{\xi_i\}$ sono gli autovalori di A^{-1} con

$$|\xi_1| > |\xi_2| \geq |\xi_3| \geq \ldots \geq |\xi_n|$$

allora il metodo delle potenze inverse calcola un'approssimazione di ξ_1 e di un suo autoversore x. Si osserva subito che se $A^{-1}x = \xi_i x$ (con $\xi_i \neq 0$) allora moltiplicando ambo membri per ${\xi_i}^{-1}A$ si ottiene leggendo da destra a sinistra $Ax = \xi_i^{-1}x$ cioè ξ_i^{-1} è un autovalore di A e x è non solo autovettore di A^{-1} relativo all'autovalore ξ_i, ma pure autovettore di A relativo all'autovalore ξ_i^{-1}. Conseguentemente se ξ_1 è l'autovalore di massimo modulo di A^{-1} e λ_n è l'autovalore di minimo modulo di A si ha $\lambda_n = \xi_i^{-1}$ e che

$$A^{-1}x = \xi_1 x \Rightarrow Ax = \xi_1^{-1}x = \lambda_n x$$

Notiamo che il metodo delle potenze inverse, calcola $\xi_1 = \lambda_n^{-1}$ e il relativo autovettore x. Per ottenere λ_n viene naturale calcolare ξ_1^{-1}, ma usualmente essendo x autovettore di A relativo a λ_n si preferisce per questioni numeriche calcolare λ_n via coefficente di Rayleigh

$$\rho(x, A) := \frac{(x, Ax)}{(x, x)}.$$

In generale, fissato $\mu \in \mathbb{C}$ è possibile calcolare l'autovalore λ più vicino a μ considerando il seguente pseudocodice [7, p.181]:

$$\begin{aligned} (A - \mu I)\, z_k &= q_{k-1} && (13)\\ q_k &= z_k / \|z_k\|_2 && (14)\\ \sigma_k &= (q_k)^H A q_k && (15) \end{aligned}$$

Ricordiamo che se λ è autovalore di A allora

$$Ax = \lambda x \Rightarrow (A - \mu I)x = \lambda x - \mu x = (\lambda - \mu)x$$

e quindi $\lambda - \mu$ è autovalore di $A - \mu I$. Il metodo delle potenze inverse applicato a $A - \mu I$ calcola il minimo autovalore $\{\sigma_i = \lambda_i - \mu\}$ in modulo di $A - \mu I$ cioè il σ_i che rende minimo il valore di $|\sigma_i| = |\lambda_i - \mu|$. Quindi essendo $\lambda_i = \sigma_i - \mu$ si ottiene pure il λ_i più vicino a μ.

Per versioni piú sofisticate con shift (o in norma infinito invece che in norma 2) si confronti [2, p.379].

Una versione di base `invpower` del metodo delle potenze inverse [7, p.184] è

```
function [lambda, x, niter, err]=invpower(A,z0,mu,toll,nmax)

% DATO UN VALORE mu, SI CALCOLA L'AUTOVALORE "lambda_mu" PIU' VICINO A mu.
```

```
% INPUT:
% A    : MATRICE DI CUI VOGLIAMO CALCOLARE L'AUTOVALORE "lambda_mu".
% z0   : VETTORE INIZIALE (NON NULLO).
% mu   : VALORE DI CUI VOGLIAMO CALCOLARE L'AUTOVALORE PIU' VICINO.
% toll: TOLLERANZA.
% nmax: NUMERO MASSIMO DI ITERAZIONI.
%
% OUTPUT:
% lambda : VETTORE DELLE APPROSSIMAZIONI DELL'AUTOVALORE DI MINIMO MODULO.
% x      : AUTOVETTORE RELATIVO ALL'AUTOVALORE DI MINIMO MODULO.
% niter  : NUMERO DI ITERAZIONI.
% err    : VETTORE DEI RESIDUI PESATI RELATIVI A "lambda".
%
% TRATTO DA QUARTERONI-SALERI, "MATEMATICA NUMERICA", p. 184.
%

n=max(size(A)); M=A-mu*eye(n); [L,U,P]=lu(M);
q=z0/norm(z0); q2=q'; err=[]; lambda=[];
res=toll+1; niter=0;
while (res >= toll & niter <= nmax)
    niter=niter+1; b=P*q; y=L\b; z=U\y;
    q=z/norm(z); z=A*q; lam=q'*z;
    b=q2'; y=U'\b; w=L'\y;
    q2=(P'*w)'; q2=q2/norm(q2); costheta=abs(q2*q);
    if (costheta > 5e-2)
        res=norm(z-lam*q)/costheta; err=[err; res]; lambda=[lambda; lam];
    else
        disp('\n \t [ATTENZIONE]: AUTOVALORE MULTIPLO'); break;
    end
    x=q;
end
```

Forniamo ora alcune spiegazioni del codice in `invpower`.

1. Per risolvere il sistema lineare in 13, si effettua una fattorizzazione $PM = LU$ della matrice $M = A - \mu I$;
2. All'interno del ciclo while, nella prima riga si calcola z_k, mentre nella successiva un suo versore q_k, e σ_k è immagazzinato in `lam`;
3. Similmente al metodo diretto si effettua il prodotto scalare di un'autovalore sinistro con uno destro.

4 Una demo sul metodo delle potenze e delle potenze inverse

Vogliamo ora applicare il metodo delle potenze e delle potenze inverse per il calcolo di alcuni autovalori di una matrice di Poisson. Salviamo nel file `demoautovalori1.m` il seguente codice Matlab/Octave:

```
siz=5;
A = makefish(siz);
eigs=eig(A);
abs_eigs=abs(eigs);
min_eig=min(abs(eigs));
max_eig=max(abs(eigs));
mu=0;

x0=ones(size(A,1),1); nmax=50; toll=10^(-11);

% METODO DELLE POTENZE.
[lambda1, x1, niter, hist_dp]=powerm(A,x0,toll,nmax);
lambdamax=lambda1(length(lambda1));
fprintf('\n \t [MAGGIOR AUTOVALORE (IN MODULO)]');
fprintf('\n \t [CALCOLATO]: %5.15f [ESATTO]: %5.15f',lambdamax,max_eig);
res=norm(A*x1-lambdamax*x1);
fprintf('\n \t [ABS.ERR.]: %2.2e [RES]: %2.2e',
abs(lambdamax-max_eig),res);


% METODO DELLE POTENZE INVERSE.
[lambda, x, niter, hist_ip]=invpower(A,x0,mu,toll,nmax);
lambdamin=lambda(length(lambda));
fprintf('\n \n \t [MINOR AUTOVALORE (IN MODULO)  ]');
fprintf('\n \t [CALCOLATO]: %5.15f [ESATTO]: %5.15f',lambdamin,min_eig);
res=norm(A*x-lambdamin*x);
fprintf('\n \t [ABS.ERR.]: %2.2e [RES]: %2.2e',
abs(lambdamin-min_eig),res);

% METODO DELLE POTENZE INVERSE PER "MU" NON NULLO.
mu=5.9;
[minval,minindex]=min(abs(eigs-mu));
minmu_eig=eigs(minindex);
[lambdamu, x, niter, hist_ipmu]=invpower(A,x0,mu,toll,nmax);
lambdamumin=lambdamu(length(lambdamu));
fprintf('\n \n \t [MINOR AUTOVALORE (IN MODULO)  ]');
fprintf('\n \t [CALCOLATO]: %5.15f [ESATTO]:
%5.15f',lambdamumin,minmu_eig);
res=norm(A*x-lambdamumin*x);
fprintf('\n \t [MU]: %5.5f [ABS.ERR.]: %2.2e [RES]:
%2.2e',mu,abs(lambdamumin-minmu_eig),res);
```

```
% METODO DELLE POTENZE INVERSE PER "MU" NON NULLO.
mu2=2.8;
[minval,minindex]=min(abs(eigs-mu2));
minmu2_eig=eigs(minindex);
[lambdamu2, x, niter, hist_ipmu2]=invpower(A,x0,mu2,toll,nmax);
lambdamu2min=lambdamu2(length(lambdamu2));
fprintf('\n \n \t [MINOR AUTOVALORE (IN MODULO)  ]');
fprintf('\n \t [CALCOLATO]: %5.15f [ESATTO]:
%5.15f',lambdamu2min,minmu2_eig);
res=norm(A*x-lambdamu2min*x);
fprintf('\n \t [MU]: %5.5f [ABS.ERR.]: %2.2e [RES]:
%2.2e',mu2,abs(lambdamu2min-minmu2_eig),res);

hold off;
semilogy(1:length(hist_dp),hist_dp,'k-*'); hold on;
semilogy(1:length(hist_ip),hist_ip,'b-o'); hold on;
semilogy(1:length(hist_ipmu),hist_ipmu,'m-d'); hold on;
semilogy(1:length(hist_ipmu2),hist_ipmu2,'r-v');

% legend('DIR','INV','MU1','MU2');
```

La demo effettua quanto segue.

1. Definita la matrice A, i suoi autovalori vengono calcolati con `eig`, così da poter vedere il comportamento dei vari metodi.
2. Successivamente testiamo il metodo delle potenze per il calcolo dell'autovalore di massimo modulo.
3. In seguito calcoliamo l'autovalore di minimo modulo e quelli più vicini a $\mu = 5.9$ e $\mu = 2.8$.
4. Alla fine plottiamo in scala semilogaritmica, iterazione per iterazione, il residuo (pesato).
5. La parte relativa a

   ```
   % legend('DIR','INV','MU1','MU2');
   ```

   è commentata, in quanto il comando `legend` non è implementato nelle vecchie versioni di Octave. Per usare la legenda, basta togliere il commento.

Lanciato il programma `demoautovalori1`, otteniamo

```
>> demoautovalori1
```

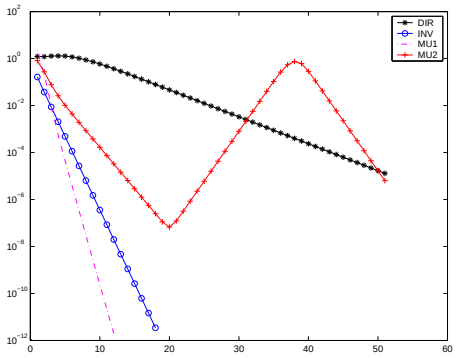

Figure 1: Grafico che illustra il residuo per iterazione del metodo delle potenze diretto e nelle versioni di Wielandt.

```
[MAGGIOR AUTOVALORE (IN MODULO)]
[CALCOLATO]: 7.464101615040614 [ESATTO]: 7.464101615137755
[ABS.ERR.]: 9.71e-011 [RES]: 1.30e-005

[MINOR AUTOVALORE (IN MODULO)  ]
[CALCOLATO]: 0.535898384862246 [ESATTO]: 0.535898384862247
[ABS.ERR.]: 1.67e-015 [RES]: 3.46e-012

[MINOR AUTOVALORE (IN MODULO)  ]
[CALCOLATO]: 5.732050807568878 [ESATTO]: 6.000000000000000
[MU]: 5.90000 [ABS.ERR.]: 2.68e-001 [RES]: 1.96e-012

[MINOR AUTOVALORE (IN MODULO)  ]
[CALCOLATO]: 2.999999999999130 [ESATTO]: 3.000000000000000
[MU]: 2.80000 [ABS.ERR.]: 8.71e-013 [RES]: 7.98e-007
>>
```

Come errore assoluto abbiamo calcolato $e_a := |\lambda - \lambda_c|$ dove λ è l'autovalore esatto mentre λ_c è quello calcolato. Quindi abbiamo calcolato la norma 2 del residuo $r_2 := \|Ax_c - \lambda_c x_c\|$ dove x_c e l'autovettore calcolato dal metodo delle potenze (diretto o inverso) relativo all'autovalore $\lambda \approx \lambda_c$.
Osserviamo che può esserci molta differenza tra e_a e r_2. Nel metodo delle potenze dirette, e nel calcolo dell'autovalore più vicino a $\mu = 2.8$ si ha infatti $e_a \ll r_2$, mentre nel calcolo dell'autovalore più vicino a $\mu = 5.9$ accade il contrario.

5 Il metodo QR

Sia A una matrice quadrata di ordine n. Utilizzando il metodo di Householder è possibile fattorizzare la matrice A come prodotto di due matrici Q ed R con Q unitaria (cioè $Q^T * Q = Q * Q^T = I$) ed R triangolare superiore.

Citiamo alcune cose:

1. La matrice A ha quale sola particolarità di essere quadrata.
2. Come osservato in [1] p. 614, tale fattorizzazione non è unica (i segni delle componenti sulla diagonale della matrice A possono essere scelti arbitrariamente).
3. La routine Matlab `QR` effettua tale fattorizzazione.
4. Se la matrice H è simile a K (cioè esiste una matrice non singolare S tale che $H = S^{-1}KS$) allora H e K hanno gli stessi autovalori. Si può vedere che la relazione di similitudine è transitiva.

Il metodo QR venne pubblicato nel 1961 da Francis e successivamente implementato in EISPACK. Ci limiteremo a considerare versioni di base del metodo.
Sia

$$A_0 = A = Q_0 R_0$$

e si ponga

$$A_1 := R_0 Q_0.$$

Poichè

$$Q_0 A_1 Q_0^T = Q_0 A_1 Q_0^T = Q_0 R_0 Q_0 Q_0^T = A_0$$

la matrice A_1 è simile ad A_0 (si ponga $S = Q_0^{-1} = Q_0^T$) e quindi ha gli stessi autovalori.
Sia quindi in generale

$$A_k = Q_k R_k$$
$$A_{k+1} = R_k Q_k.$$

Per le stesse motivazioni A_{k+1} è simile ad A_k, e per transitività ad A_0. Quindi A_{k+1} ha gli stessi autovalori di A_0.
Per la convergenza del metodo citiamo il seguente risultato (cf. [3], p. 393).

Teorema 5.1 *Sia V^* la matrice formata dagli autovettori di sinistra della matrice A. Se tutti i minori principali di V sono diversi da* 0 *e se gli autovettori di A verificano la condizione*

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0. \tag{16}$$

allora la matrice $A_{k+1} = Q_k^T A Q_k$ tende a una forma triangolare superiore e $(A_{k+1})_{ii}$ converge verso λ_i.

Se la condizione (16) non è verificata si può dimostrare che la successione $\{A_k\}$ tende a una forma triangolare a blocchi. Per versioni piú sofisticate come il metodo QR con shift, si veda [3], p. 394.
Nelle implementazioni si calcola con un metodo scoperto da Householder (ma esiste un metodo alternativo dovuto a Givens) una matrice di Hessenberg T

$$T = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,n} \\ 0 & a_{3,2} & a_{3,3} & \ldots & a_{3,n} \\ 0 & 0 & a_{4,3} & \ldots & a_{4,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

simile ad A ed in seguito si applica il metodo QR relativamente alla matrice T. Se A è simmetrica la matrice T risulta tridiagonale simmetrica.
Per inciso, si può dimostrare che la velocità di convergenza dipende dal rapporto

$$\max_{1\le i\le n-1} \max \frac{|\lambda_{i+1}|}{|\lambda_i|}.$$

Una versione di base del metodo QR è la seguente. Si salvi il files `houshess.m` che implementa la trasformazione per similitudine di A in una matrice di Hessenberg

```
function [H,Q]=houshess(A)

% REDUCTION OF A MATRIX TO A SIMILAR HESSENBERG ONE.
% SEE QUARTERONI, SACCO, SALERI P. 192.

n=max(size(A)); Q=eye(n); H=A;

for k=1:(n-2)
    [v,beta]=vhouse(H(k+1:n,k)); I=eye(k); N=zeros(k,n-k);
    m=length(v); R=eye(m)-beta*v*v'; H(k+1:n,k:n)=R*H(k+1:n,k:n);
    H(1:n,k+1:n)=H(1:n,k+1:n)*R; P=[I,N; N',R]; Q=Q*P;
end
```

quindi `QRbasicmethod.m` che calcola la matrice triangolare T relativa a QR:

```
function [T,hist]=QRbasicmethod(T_input,maxit)

% QR METHOD FOR A SYMMETRIC TRIDIAGONAL MATRIX "T_input".

T=T_input;
hist=sort(diag(T));

for index=1:maxit
    [Q,R]=qr(T);
    T=R*Q;                      % NEW SIMILAR MATRIX.
    hist=[hist sort(diag(T))]; % IT STORES THE DIAGONAL ELEMENTS
                                % OF THE "index" ITERATION.
end
```

Il codice `houshess.m` utilizza `vhouse.m`

```
function [v,beta]=vhouse(x)

% BUILDING HOUSEHOLDER VECTOR.
% SEE QUARTERONI, SACCO, SALERI P. 197.

n=length(x); x=x/norm(x); s=x(2:n)'*x(2:n); v=[1; x(2:n)];
```

```
if (s==0)
    beta=0;
else
    mu=sqrt(x(1)^2+s);
    if (x(1) <= 0)
        v(1)=x(1)-mu;
    else
        v(1)=-s/(x(1)+mu);
    end
    beta=2*v(1)^2/(s+v(1)^2);
    v=v/v(1);
end
```

6 Sugli autovalori della matrice di Poisson

In questa sezione desideriamo calcolare gli autovalori della matrice di Poisson che si ottiene dal comando `makefish`. Più precisamente in `demoQR.m` scriviamo il seguente codice per il calcolo degli autovalori della matrice di Poisson `makefish(2)`, mostrando come la matrice di partenza A viene trasformata in una matrice di Hessenberg H da cui si calcola la matrice T triangolare superiore prodotta dalle iterazioni di QR:

```
format short

maxit=100;
siz=2;

A = makefish(siz);
eigs=eig(A);

[H,Q]=houshess(A);

[T,hist]=QRbasicmethod(H,maxit);

A
H
T
eig(A)
```

Otteniamo come risultato

```
>> demoQR
A =

     4    -1    -1     0
    -1     4     0    -1
    -1     0     4    -1
```

```
     0    -1    -1     4

H =

    4.0000    1.4142    0.0000   -0.0000
    1.4142    4.0000    1.4142   -0.0000
   -0.0000    1.4142    4.0000   -0.0000
         0   -0.0000   -0.0000    4.0000

T =

    6.0000    0.0000   -0.0000   -0.0000
    0.0000    4.0000    0.0000   -0.0000
   -0.0000   -0.0000    4.0000    0.0000
         0   -0.0000    0.0000    2.0000

ans =

    2.0000
    4.0000
    4.0000
    6.0000

>>
```

In cui si vede che gli autovalori, come affermato dalla teoria, sono le componenti diagonali della matrice T. Lanciare il codice `demoQR` relativamente alla matrice `makefish(3)` (e non `makefish(2)`). A tal proposito modificare la variabile `siz`.

7 Alcuni esempi ed esercizi

1. Calcolare con il metodo QR gli autovalori della matrice di Hilbert di ordine 5 (ricordare il comando `eig(hilb(5))`) e confrontarli con quelli prodotti dal metodo QR. Applicare il metodo delle potenze per il calcolo dell'autovalore di modulo massimo e il metodo di Wielandt per quelle di modulo minimo.

2. Applicare il metodo QR per il calcolo degli autovalori di

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

   E' *a blocchi* la matrice generata dal metodo QR? E se cosí non fosse come ovviare al problema? (cf. [3], p. 393). Suggerimento: ricordarsi come calcolare gli autovalori di una matrice di ordine 2 via equazioni di secondo grado.

3. Applicare il metodo QR per il calcolo degli autovalori di

$$A = \begin{pmatrix} 3 & 17 & -37 & 18 & 40 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

   E' la matrice generata dal metodo QR a blocchi? E se cosí non fosse come ovviare al problema? (cf. [7], p. 195).

4. Applicare il metodo delle potenze per il calcolo dell'autovalore di massimo modulo di

$$A(\alpha) = \begin{pmatrix} \alpha & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{pmatrix}$$

   Confrontare i risultati ottenuti con quelli di `eig(A)` per $\alpha = 30, -30$ plottando il grafico dell'errore tra l'approssimazione della soluzione fornita iterazione per iterazione dal metodo delle potenze con quella di `max(abs(eig(A)))` (cf. [8], p.133).

5. Si dica se è possibile applicare (sperimentarlo!) il metodo delle potenze per il calcolo dell'autovalore di massimo modulo di

$$A = \begin{pmatrix} 1/3 & 2/3 & 2 & 3 \\ 1 & 0 & -1 & 2 \\ 0 & 0 & -5/3 & -2/3 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

   Dare una spiegazione di tale fenomeno (cf. [8], p. 141, esercizio 6.5).

8 Facoltativo: Metodo QR: alcune considerazioni

Sia A una matrice quadrata di ordine n e siano da calcolare tutti i suoi autovalori $\{\lambda_i\}$. In precedenza abbiamo visto che é possibile trasformare per similitudine la matrice A in una matrice di Hessenberg T

$$T = \begin{pmatrix} a_{1,1} & a_{2,1} & a_{3,1} & \dots & a_{n,1} \\ a_{1,2} & a_{2,2} & a_{3,2} & \dots & a_{n,2} \\ 0 & a_{2,3} & a_{3,3} & \dots & a_{n,3} \\ 0 & 0 & a_{3,4} & \dots & a_{n,4} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

Essendo T simile ad A, gli autovalori di T sono gli stessi di A. Si é osservato che una versione di base del metodo QR può incontrare problemi nel convergere qualora esistano 2 autovalori distinti di uguale metodo.

Questo é ad esempio il caso della matrice

$$\begin{pmatrix} 3 & 17 & -37 & 18 & 40 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

che ha due valori complessi coniugati

$$\lambda \approx 1.27437503036000 \pm 1.03039271164378\, i.$$

Affrontiamo il caso della matrice di Hessenberg di ordine 3 (cf. [3], p. 393)

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

Si vede facilmente che alla 25-sima iterazione del metodo QR (in una sua versione di base) corrisponde la matrice

$$A_{25} = \begin{pmatrix} 2.000 & 1.069 & 0.926 \\ 0.000 & -0.500 & 0.866 \\ 0 & -0.866 & -0.500 \end{pmatrix}$$

(abbiamo scritto per semplicità 3 cifre decimali delle componenti di A_{25}). Risulta chiaro che quindi il metodo QR non fornisce gli autovalori di A, in quanto non tende a una matrice triangolare superiore. Il problema come in precedenza é dovuto alla presenza di 2 autovalori complessi coniugati

$$\lambda \approx -0.5 \pm 0.8660\, i.$$

Sorge quindi naturale la questione sul come trattare tali casi.

Una seconda osservazione nasce spontanea dal teorema di convergenza del metodo QR. Questa volta consideriamo una sua versione semplificata (cf. [1], p. 625)

Teorema 8.1 *Sia A una matrice di ordine n e supponiamo che i suoi autovalori $\{\lambda_i\}_{i=1,\dots,n}$ soddisfino*

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$$

Allora le iterate R_m del metodo QR convergono a una matrice triangolare superiore T che contiene gli autovalori $\{\lambda_i\}_{i=1,\dots,n}$ nelle componenti della diagonale. Se A é simmetrica converge a una matrice diagonale. Per quanto riguarda la velocità di convergenza,

$$\|D - A_m\| \leq c \max_i \frac{|\lambda_{i+1}|}{|\lambda_i|} \tag{17}$$

Sono necessarie alcune osservazioni. Rispetto al teorema introdotto in precedenza viene a mancare un'ipotesi sugli autovettori sinistri di A che aveva quale effetto il fatto che gli autovalori sulla diagonale D di A erano ordinati (decrescentemente

rispetto al valore delle componenti in modulo). Mentre nel teorema precedente si diceva che le matrici A_k, iterazioni di QR, tendevano a una matrice tridiagonale superiore, qui si parla di R_m ovvero quella matrice triangolare superiore per cui $A_m = Q_m R_m$. La cosa comunque basilare é che a *guidare* la velocità di convergenza é il termine

$$\rho = \max_i \frac{|\lambda_{i+1}|}{|\lambda_i|}. \tag{18}$$

Come sfruttare questo fatto per migliorare le proprietà di convergenza del metodo?

9 Facoltativo: Metodo QR con shift

la velocità di convergenza dipende dal rapporto ρ in (18). Se é vicino a 1 la convergenza può essere lenta o perfino non sussistere (si vedano gli esempi della sezione precedente). Per risolvere questi problemi si utilizza una tecnica detta dello shift. Sia μ un numero (anche complesso!) che approssima meglio degli altri un autovalore e si consideri la successione $\{A_k\}$ generata da

$$\begin{cases} A_k - \mu I = Q_k R_k \\ A_{k+1} = R_k Q_k + \mu I_n \end{cases} \tag{19}$$

dove al solito I_n é la matrice identica di ordine n, e $A_0 := A$.
Osserviamo che essendo Q_k matrici unitarie cioé $Q_K^H Q_K = Q_K Q_K^H = I_n$ si ha

$$Q_K A_{k+1} Q_K^H = Q_K (R_k Q_k + \mu I_n) Q_K^H = Q_k R_K + \mu I_n = A_k$$

per cui si evince che A_{k+1} ed A_k sono simili. Conseguentemente A_k é simile ad $A_0 = A - \mu I_n$ e i suoi autovalori sono $\{\lambda_i - \mu\}_{i=1,\dots,n}$ da cui si ottengono facilmente $\{\lambda_i\}_{i=1,\dots,n}$.
Esistono diverse varianti di questa tecnica.

9.1 Variante 1

In una prima (cf. [2], p. 363) si pone

$$\mu_k = (a_k)_{n,n}\,,\ k = 1,2,\dots \tag{20}$$

e si considera il metodo

$$\begin{cases} A_k - \mu_k I = Q_k R_k \\ A_{k+1} = R_k Q_k + \mu_k I_n \\ A_0 = A \end{cases} \tag{21}$$

Nel caso di matrici hermitiane (cioé $A = A^H$) la convergenza a zero di $(a_k)_{n,n-1}$ é del terzo ordine. Una volta che $|(a_k)_{n,n-1}| < \epsilon$, dove ϵ é una tolleranza prefissata, si procede riapplicando lo stesso metodo alla sottomatrice $\hat{A}$ composta dalle prime $n-1$ righe e colonne di A_k. Poiché $|(a_k)_{n,n-1}| \approx 0$, e gli autovalori di una matrice diagonale a blocchi sono l'unione degli autovalori di ogni blocco, gli autovalori di A sono l'unione di quelli di $\hat{A}$ con $(a_k)_{n,n}$.

9.2 Variante 2 (di Wilkinson)

Nel caso in cui $|\lambda_{n-1}| = |\lambda_n|$ si procede come in precedenza scegliendo μ_k l'autovalore della sottomatrice

$$\begin{pmatrix} a_{n-1,n-1} & a_{n,n} \\ a_{n,n-1} & a_{n,n} \end{pmatrix} \tag{22}$$

che é piú vicino ad $(a_k)_{n,n}$. Si noti che in questo caso, anche se la matrice A ha coefficienti reali, l'utilizzazione dello shift può portare ad una matrice ad elementi complessi, con un aumento del costo computazionale.

10 Online

Sul calcolo degli autovalori di una matrice è possibile trovare online e gratuitamente del software per Matlab.

1. Dal sito

   `http://mox.polimi.it/~fausal/matlab.html`

   si possono scaricare i files `eigpower.m` e `invshift.m` che implementano i metodi delle potenze. e `qrbasis.m` una versione di base del metodo QR.

2. Vario software di algebra lineare è presente al sito

   `http://www.math.sc.edu/~meade/math706/MATLAB-files/index.html`

   in particolare le versioni con shift di QR citate poco sopra.

3. Estremamente interessante la homepage di Y. Saad

   `http://www-users.cs.umn.edu/~saad/books.html`

   in cui si trovano gratuitamente manuali sul calcolo di autovalori e risoluzione di sistemi lineari.

References

[1] K. Atkinson, *Introduction to Numerical Analysis*, Wiley, 1989.

[2] D. Bini, M. Capovani e O. Menchi, *Metodi numerici per l'algebra lineare*, Zanichelli, 1988.

[3] V. Comincioli, *Analisi Numerica, metodi modelli applicazioni*, Mc Graw-Hill, 1990.

[4] S.D. Conte e C. de Boor, *Elementary Numerical Analysis, 3rd Edition*, Mc Graw-Hill, 1980.

[5] The MathWorks Inc., *Numerical Computing with Matlab*, `http://www.mathworks.com/moler`.

[6] Netlib, `http://www.netlib.org/templates/matlab/`.

[7] A. Quarteroni, R. Sacco, F. Saleri, *Matematica numerica*, 2001.

[8] A. Quarteroni e F. Saleri, *Introduzione al calcolo scientifico*, Springer Verlag, 2006.

[9] A. Suli e D. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.