Calcolo di autovalori e autovettori

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

29 aprile 2019

Metodo delle potenze

Metodo (Potenze)

Sia $t_0 \in \mathbb{R}^n$ *definito da*

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

Il metodo delle potenze genera la successione

$$\begin{aligned} y_0 &= t_0 \\ y_k &= A y_{k-1}, \ k = 1, 2, \ldots \end{aligned}$$

Nota.

E' noto un teorema di convergenza per matrici diagonalizzabili, qualora gli autovettori $\{\lambda_j\}$ *siano tali che* $|\lambda_1| > |\lambda_k|$ *per ogni* $k > 1$ *e il vettore iniziale non sia parallelo all'autovettore* u_1 *relativo all'autovalore* λ_1.

Il metodo delle potenze in Matlab

Partiamo con una versione semplice power_basic del metodo delle potenze

```
function [lambda1, x1, niter, err]=power_basic(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);
    niter=niter+1; res=norm(z-lam*q)/costheta;
    err=[err; res]; lambda1=[lambda1; lam];
end
```

Il metodo delle potenze in Matlab

Qualche nota

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

Il metodo delle potenze in Matlab

- nel ciclo `while`, `q` è un'approssimazione di un autoversore relativo a $\lambda_{\max}$, mentre `lam` di $\lambda_{\max}$;
- 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. [2, p.180])

Il metodo delle potenze in Matlab: esempio I

Testiamo il codice per il calcolo dell'autovalore di massimo modulo di

$$A = \begin{pmatrix} -15.5 & 7.5 & 1.5 \\ -51 & 25 & 3 \\ -25.5 & 7.5 & 11.5 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix} \cdot \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix}^{-1} \quad (1)$$

La matrice A è diagonalizzabile e ha autovalori 10, 10, 1. Si può vedere che una base di autovettori relativa agli autovalori 10, 10, 1 è composta da $(1,2,7)$, $(2,5,9)$, $(3,6,3)$. Quale vettore iniziale del metodo delle potenze consideriamo

$$z_0 = (1,1,1) = (7/6)\cdot(1,2,7) - 1\cdot(2,5,9) + (11/18)\cdot(3,6,3)$$

e quindi il metodo delle potenze applicato ad A, e avente quale punto iniziale z_0 può essere utilizzato per il calcolo dell'autovalore di massimo modulo di A, poichè $\alpha_1 = 7/6 \neq 0$.

Il metodo delle potenze in Matlab: esempio I

Dalla shell di Matlab/Octave:

```
>> S=[1 2 3; 2 5 6; 7 9 3];
>> D=diag([10 10 1]);
>> A=S*D*inv(S)
A =
   -15.5000    7.5000    1.5000
   -51.0000   25.0000    3.0000
   -25.5000    7.5000   11.5000
>> z0=[1 1 1]';
>> toll=10^(-8);
>> nmax=10;
>> format short e;
```

Il metodo delle potenze in Matlab: esempio I

```
>> [lambda1, x1, niter, err]=power_basic(A,z0,toll,nmax)
lambda1 =
   1.1587e+001
   1.0138e+001
   1.0014e+001
   1.0001e+001
    ...
   1.0000e+001
   1.0000e+001
   1.0000e+001
x1 =
  -2.8583e-001
  -9.1466e-001
  -2.8583e-001
niter =
     10
err =
   2.2466e+000
   2.1028e-001
   2.0934e-002
   2.0925e-003
   ...
   2.0924e-007
   2.0924e-008
   2.0924e-009
>>
```

Il metodo delle potenze in Matlab: esempio I

La convergenza è abbastanza veloce come si vede dalla quantità `err`, che consiste in un particolare residuo pesato.
Una questione sorge spontanea. Cosa sarebbe successo se avessimo utilizzato l'algoritmo senza normalizzazione come ad esempio power_method definito da

```
function [lambda,v]=power_method(A,x0,maxit)

v=x0;

for index=1:maxit
    v_old=v;
    v=A*v_old;
    lambda=(v_old'*v)/(v_old'*v_old);
end
```

Il metodo delle potenze in Matlab: esempio I

Proviamo il test, facendo iterare il metodo prima 5, poi 100 volte e alla fine 1000 volte (si noti il settaggio della variabile `maxit` relativa al numero di iterazioni da compiere):

```
>> x0=[1 1 1]'
x0 =
     1
     1
     1
>> A=[-15.5 7.5 1.5; -51 25 3; -25.5 7.5 11.5]
A =
  -15.5000    7.5000    1.5000
  -51.0000   25.0000    3.0000
  -25.5000    7.5000   11.5000
>> [lambda,v]=power_method(A,x0,5)
lambda =
   10.0014
v =
  1.0e+005 *
   -0.8333
   -2.6666
   -0.8333
>>
```

Il metodo delle potenze in Matlab: esempio I

```
>> [lambda,v]=power_method(A,x0,100)
lambda =
    10.0000
v =
   1.0e+100 *
    -0.8333
    -2.6667
    -0.8333
>> [lambda,v]=power_method(A,x0,1000)
lambda =
    NaN
v =
    NaN
    NaN
    NaN
>>
```

Il metodo delle potenze in Matlab: esempio I

La ragione è semplice. Per k relativamente piccolo si ha $A \cdot t_k \approx 10 \cdot t_k$ e quindi per $s \geq k$

$$t_s \approx A^{s-k} \cdot t_k \approx 10 \cdot A^{s-k-1} \cdot t_k \approx \cdots \approx 10^{s-k} \cdot t_k$$

da cui

$$\|t_s\|_2 \approx 10^{s-k} \cdot \|t_k\|_2$$

spiegando quindi perchè si possano avere problemi di overflow applicando l'algoritmo di base.

Il metodo delle potenze in Matlab: esempio II

Proviamo un test diverso, questa volta con la matrice (diagonalizzabile)

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

avente autovalori $\lambda_1 = 1$ e $\lambda_2 = -1$ e autovettori linearmente indipendenti $(1, 0)$, $(-1, 1)$. Quale vettore iniziale poniamo

$$x_0 = (1, 3) = 4 \cdot (1, 0) + 3 \cdot (-1, 1)$$

e quindi il metodo delle potenze applicato ad A, partendo da x_0 può essere sicuramente applicato. D'altra parte dubitiamo converga in quanto $|\lambda_1| = |\lambda_2| = 1$ pur essendo $\lambda_1 \neq \lambda_2$.

Il metodo delle potenze in Matlab: esempio II

Dalla shell di Matlab/Octave:

```
>> A=[1 2; 0 -1]
A =
     1     2
     0    -1
>> [lambda1, x1, niter, err]=power_basic(A,[1; 3],10^(-8),15)
lambda1 =
  -3.4483e-002
  -2.0000e-001
  ...
  -3.4483e-002
  -2.0000e-001
  -3.4483e-002
  -2.0000e-001
x1 =
   3.1623e-001
   9.4868e-001
niter =
    16
err =
   4.4567e-001
   2.4000e+000
   ...
   4.4567e-001
   2.4000e+000
   4.4567e-001
   2.4000e+000
>>
```

Il metodo delle potenze in Matlab: esempio III

Vediamo il caso della matrice diagonalizzabile (autovalori distinti!)

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 10 \end{pmatrix},$$

in cui il metodo funziona rapidamente ($\lambda_{\max} = 10$).

```
>> A=[1 2; 0 10]; [lambda1, x1, niter, err]=power_basic(A,[1; 3],10^(-8),15)
lambda1 =
   9.9779e+000
   9.9979e+000
   9.9998e+000
   ...
   1.0000e+001
   1.0000e+001
   1.0000e+001
x1 =
   2.1693e-001
   9.7619e-001
niter =
     8
err =
   9.6726e-002
   9.7529e-003
   9.7610e-004
   ...
   9.7619e-007
   9.7619e-008
   9.7619e-009
>>
```

Il metodo delle potenze inverse in Matlab

Una versione di base invpower del metodo delle potenze inverse [2, 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.
%
```

Il metodo delle potenze inverse in Matlab

```
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
```

Il metodo delle potenze inverse in Matlab

Forniamo ora alcune spiegazioni del codice in `invpower`.

- Per risolvere il sistema lineare, si effettua una fattorizzazione $PM = LU$ della matrice $M = A - \mu I$;
- 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`;
- Similmente al metodo diretto si effettua il prodotto scalare di un'autovalore sinistro con uno destro.

Il metodo delle potenze inverse in Matlab: esempio I

Applichiamo il metodo delle potenze inverse per il calcolo dell'autovalore più piccolo in modulo della matrice

$$\begin{aligned} A &= \begin{pmatrix} -15.5 & 7.5 & 1.5 \\ -51 & 25 & 3 \\ -25.5 & 7.5 & 11.5 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix} \cdot \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix}^{-1} \end{aligned} \tag{2}$$

Come visto la matrice A è quindi diagonalizzabile, ha autovalori 10, 10, 1 e relativi autovettori è $(1,2,7)$, $(2,5,9)$, $(3,6,3)$ formanti una base di $\mathbb{R}^3$. Quale vettore iniziale consideriamo

$$z_0 = (1,1,1) = (7/6)\cdot(1,2,7) - 1\cdot(2,5,9) + (11/18)\cdot(3,6,3)$$

e quindi il metodo delle potenze inv. applicato ad A, e avente quale punto iniziale z_0 può essere utilizzato per il calcolo dell'autovalore di minimo modulo di A.

Il metodo delle potenze inverse in Matlab: esempio I

```
>> z0=[1;1;1]; mu=0; toll=10^(-8); nmax=10;
>> A=[-15.5 7.5 1.5; -51 25 3; -25.5 7.5 11.5];
>> [lambda, x, niter, err]=invpower(A,z0,mu,toll,nmax)
lambda =
   0.39016115351993
   0.94237563941268
   0.99426922936854
   0.99942723776656
   0.99994272692315
   0.99999427272378
   0.99999942727270
   0.99999994272728
   0.99999999427273
x =
   0.40824829053809
   0.81649658085350
   0.40824829053809
niter =
     9
err =
   0.81535216507377
   0.08358101289062
   0.00838126258396
   0.00083836078891
   0.00008383842712
   0.00000838386620
   0.00000083838685
   0.00000008383868
   0.00000000838387
>>
```

Il metodo delle potenze inverse in Matlab: esempio I

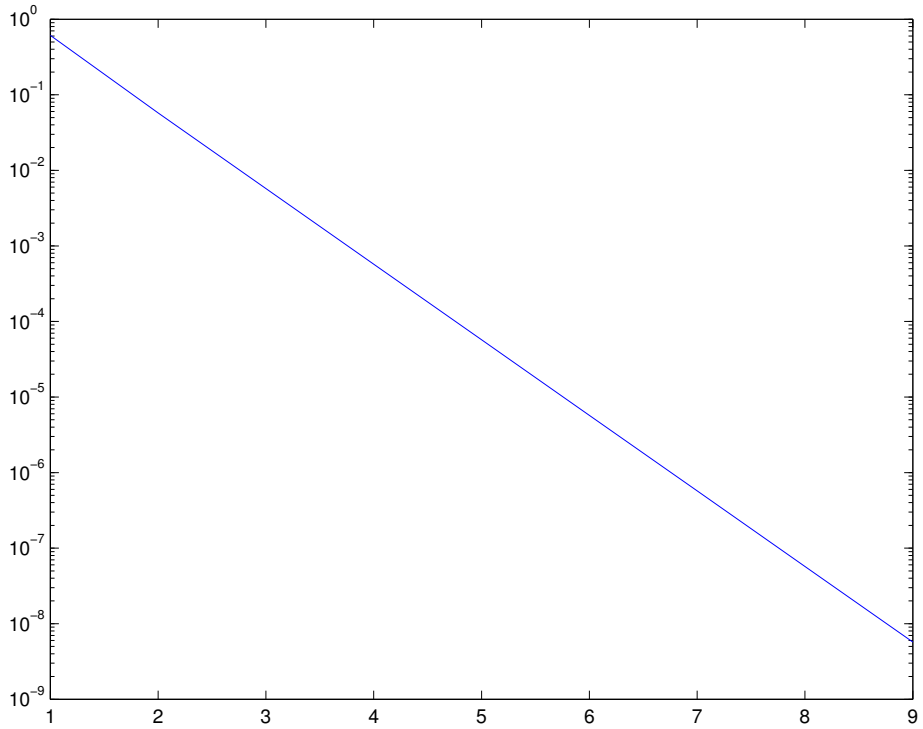

Figura: Grafico che illustra la convergenza lineare del metodo delle potenze inverse nell'esempio 1.

Il metodo delle potenze inverse in Matlab: esempio I

Per vederlo, dalla shell di Matlab/Octave calcoliamo l'errore assoluto/relativo relativo all'autovalore 1:

```
>> s=1-lambda
s =
    0.60983884648007
    0.05762436058732
    0.00573077063146
    0.00057276223344
    0.00005727307685
    0.00000572727622
    0.00000057272730
    0.00000005727272
    0.00000000572727
>> semilogy(1:length(s),s)
```

generando il grafico in scala semi-logaritmica in figura che evidentemente sottolinea la convergenza lineare.

Il metodo QR in Matlab

Nel metodo QR, data una matrice A, la si trasforma in una matrice simile in forma di Hessenberg $A_0 = T$ e quindi si eseguono le iterazioni

$$A_k = Q_k R_k, \quad A_{k+1} = R_{k+1} Q_{k+1}$$

ottenendo una successione di matrici di Hessenberg, convergenti, sotto opportune ipotesi a una matrice triangolare (a blocchi).

Il metodo QR in Matlab

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
```

Il metodo QR in Matlab

ove vhouse.m è definito da

```
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
```

Il metodo QR in Matlab

A partire da queste routines, QRbasicmethod.m determina la matrice (quasi-)triangolare (a blocchi) T descritta nel teorema di convergenza del metodo 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 non è ovviamente ottimizzato e si interrompe dopo `maxit` iterazioni, senza alcun controllo sull'errore.

Esercizio

Esercizio

- *Data la matrice di Hilbert di ordine 5, ottenibile in Matlab col comando* `hilb(5)` *si calcolino col metodo delle potenze i suoi minimi e massimi autovalori in modulo.*
- *Da questi si determini il condizionamento della matrice in norma 2 e lo si confronti con* `cond(hilb(5),2)`*. Eseguire lo stesso esercizio utilizzando il metodo QR.*

Bibliografia

Netlib,http://www.netlib.org/templates/matlab/

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