Autovalori

26 febbraio 2009

1 Calcolo degli autovalori di una matrice

Il problema del calcolo degli autovalori [16] 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 , dopo i teoremi di localizzazione di Gerschgorin, 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 [3], [22].

Una interessante applicazione è l'algoritmo di PageRank [20], utilizzato da Google per fornire i risultati migliori tra i siti web relativamente a certe parole chiave ed in prima approssimazione basato sul calcolo di un autovettore relativo all'autovalore 1 (ad esempio via metodo delle potenze) di una matrice stocastica di dimensioni *enormi* (cf. [1], [4], [5]).

2 Teoremi di Gerschgorin

In questo paragrafo mostriamo tre teoremi di localizzazione di autovalori dovuti a Gerschgorin (cf. [3, p.76], [23]).

Teorema 2.1 *(Primo teorema di Gerschgorin). Gli autovalori di una matrice A di ordine n sono tutti contenuti nell'unione dei* cerchi *di Gerschgorin*

$$K_i = \{z \in \mathbb{C} : |z - a_{i,i}| \leq \sum_{j=1, j\neq i}^{n} |a_{i,j}|\}$$

Vediamo quale esempio la matrice

$$A = \begin{pmatrix} 15 & -2 & 2 \\ 1 & 10 & -3 \\ -2 & 1 & 0 \end{pmatrix} \tag{2}$$

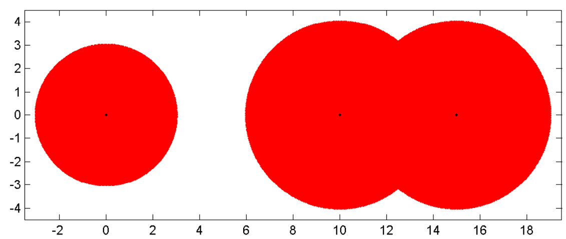

Figura 1: Cerchi di Gerschgorin della matrice A definita in (2)

Il primo teorema di Gerschgorin stabilisce che gli autovalori stanno nell'unione dei cerchi di Gerschgorin

$$\begin{aligned}
K_1 &= \{z \in \mathbb{C} : |z-15| \leq |-2|+|2| = 4\} \\
K_2 &= \{z \in \mathbb{C} : |z-10| \leq |1|+|-3| = 4\} \\
K_3 &= \{z \in \mathbb{C} : |z-0| \leq |-2|+|1| = 3\}
\end{aligned}$$

Teorema 2.2 *(Secondo teorema di Gerschgorin). Se l'unione M_1 di k cerchi di Gerschgorin è disgiunta dall'unione M_2 dei rimanenti $n-k$, allora k autovalori appartengono a M_1 e $n-k$ appartengono a M_2.*

Applicando il secondo teorema di Gerschgorin, dal confronto con la figura 2 abbiamo che 1 autovalore sta nel cerchio K_3 mentre 2 stanno nell'unione dei cerchi K_1, K_2.

Teorema 2.3 *(Terzo teorema di Gerschgorin). Se la matrice di ordine n è irriducibile e un autovalore λ sta sulla frontiera dell'unione dei cerchi di Gerschgorin, allora sta sulla frontiera di ogni cerchio di Gerschgorin.*

Questo teorema sarà utile in seguito per dimostrare che una matrice, detta di *Poisson*, è non singolare. Ricordiamo che una matrice si dice *irriducibile* se non è *riducibile* e che una matrice di ordine $n \geq 2$ è riducibile se esiste una matrice di permutazione Π e un intero k, $0 < k < n$, tale che

$$B = \Pi A \Pi^T = \begin{pmatrix} A_{1,1} & A_{1,2} \\ 0 & A_{2,2} \end{pmatrix}$$

in cui $A_{1,1} \in \mathbb{C}^{k \times k}$, $A_{12,2} \in \mathbb{C}^{(n-k)\times(n-k)}$.

Vediamo ora in Matlab quali sono effettivamente gli autovalori. si ha

```
>> A=[15 -2 2; 1 10 -3; -2 1 0]
A =
    15    -2     2
     1    10    -3
    -2     1     0
```

```
>> eig(A)
ans =
    0.5121
   14.1026
   10.3854

>>
```

a conferma di quanto stabilito dai primi due teoremi di Gerschgorin.

Nota. Cosa possiamo dire relativamente agli autovalori di A se applichiamo i teoremi di Gerschgorin ad A^T invece che ad A?

3 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

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|. \tag{3}$$

A tal proposito ricordiamo (cf. [7], p. 951) che

1. una matrice A ́diagonalizzabile se e solo se possiede n autovettori 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

$$y_0 = t_0 \tag{4}$$

$$y_k = A y_{k-1}, \; k = 1, 2, \ldots \tag{5}$$

Teorema 3.1 *Sia A è una matrice quadrata diagonalizzabile avente autovalori λ_k tali che*

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

Siano $u_k \neq 0$ autovettori relativi all'autovalore λ_k, cioè

$$A u_k = \lambda_k u_k.$$

Sia

$$y_0 = \sum_k \alpha_k u_k, \quad \alpha_1 \neq 0.$$

La successione $\{y_s\}$ definita da $y_{s+1} = Ay_s$ converge ad un vettore parallelo a x_1 e che il coefficiente di Rayleigh (relativo al prodotto scalare euclideo)

$$\rho(y_s, A) := \frac{(y_s, Ay_s)}{(y_s, y_s)} \tag{6}$$

converge a λ_1.

Dimostrazione (cf. [13, p.171]). Essendo la matrice A diagonalizzabile, esistono n autovettori u_k (relativi rispettivamente agli autovalori λ_k) che sono linearmente indipendenti e quindi formano una base di $\mathbb{R}^n$. Sia

$$y_0 = \sum_k \alpha_k u_k, \quad \alpha_1 \neq 0.$$

Essendo $Au_k = \lambda_k u_k$ abbiamo

$$y_1 = Ay_0 = A(\sum_k \alpha_k u_k) = \sum_k \alpha_k Au_k = \sum_k \alpha_k \lambda_k u_k$$

$$y_2 = Ay_1 = A(\sum_k \alpha_k \lambda_k u_k) = \sum_k \alpha_k \lambda_k Au_k = \sum_k \alpha_k \lambda_k^2 u_k$$

e più in generale

$$y_{s+1} = Ay_s = A(\sum_k \alpha_k \lambda_k^s u_k) = \sum_k \alpha_k \lambda_k^s Au_k = \sum_k \alpha_k \lambda_k^{s+1} u_k$$

Osserviamo ora che

$$\frac{y_{s+1}}{\lambda_1^s} = \sum_k \alpha_k \frac{\lambda_k^{s+1}}{\lambda_1^s} u_k \tag{7}$$

per cui essendo per $k > 1$

$$\left| \frac{\lambda_k^{s+1}}{\lambda_1^s} \right| < 1,$$

si ha

$$\lim_{s \to +\infty}, \left(\frac{\lambda_k}{\lambda_1} \right)^s = 0$$

e quindi la direzione di $\frac{y_s}{\lambda_1^s}$, che è la stessa di y_s, tende a quella dell'autovettore u_1. Per continuità il coefficiente di Rayleigh

$$\rho(y_s, A) := \frac{(y_s, Ay_s)}{(y_s, y_s)} \approx \frac{(\lambda_1^s u_1, A\lambda_1^s u_1)}{(\lambda_1^s u_1, \lambda_1^s u_1)} = \frac{(u_1, Au_1)}{(u_1 u_1)} = \lambda_1 \tag{8}$$

converge a λ_1.

Si osservi che il coefficiente di Rayleigh $\rho(\cdot, A)$ è continuo in ogni $x \neq 0$, $x \in \mathbb{R}^n$ in quanto tanto il numeratore quanto il denominatore sono funzioni polinomiali

(quadratiche) delle componenti x_k di $x = (x_k)_k \in \mathbb{R}^n$, che sono appunto continue.
$\blacksquare$
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 (4). Conseguentemente l'algoritmo del metodo delle potenze risulta

$$\begin{aligned} u_k &= A t_{k-1} && (9)\\ t_k &= \frac{u_k}{\beta_k},\ \beta_k = \|u_k\|_2 && (10)\\ l_k &= \rho(t_k, A) && (11) \end{aligned}$$

dove $\rho(t_k, A)$ è il coefficiente di Rayleigh definito in (6).

4 Il metodo delle potenze inverse

Una variante particolarmente interessante del metodo delle potenze è stata scoperta da Wielandt nel 1944 [17] ed e' particolarmente utile nel caso in cui A sia una matrice quadrata con n autovettori linearmente indipendenti,

$$|\lambda_1| \geq |\lambda_2| \geq \ldots > |\lambda_n| > 0. \tag{12}$$

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

$$\begin{aligned} A u_k &= t_{k-1} && (13)\\ t_k &= \frac{u_k}{\beta_k},\ \beta_k = \|u_k\|_2 && (14) \end{aligned}$$

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

$$\rho(t_k, A^{-1}) := \frac{(t_k, A^{-1} t_k)}{(t_k, t_k)} = \frac{(t_k, u_{k+1})}{(t_k, t_k)} \to 1/\lambda_n. \tag{15}$$

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 [12, p.181]:

$$\begin{aligned} (A - \mu I)\, z_k &= q_{k-1} && (16)\\ q_k &= z_k / \|z_k\|_2 && (17)\\ \sigma_k &= (q_k)^H A q_k && (18) \end{aligned}$$

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 $\sigma = \lambda - \mu$ in modulo di $A - \mu I$ cioè il σ che rende minimo il valore di $|\sigma| = |\lambda_i - \mu|$, dove λ_i sono gli autovalori di A. Quindi essendo $\lambda_i = \sigma_i - \mu$ si ottiene pure il λ_i più vicino a μ.

Per versioni piú sofisticate di questa tecnica detta di *shift* (o in norma infinito invece che in norma 2) si confronti [3, p.379].

Problema. Si può applicare il metodo delle potenze inverse con shift μ nel caso μ sia proprio un autovalore di A?

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 [2] 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 facilmente che la relazione di similitudine è transitiva, cioè se H_1 è simile ad H_2 e H_2 è simile ad H_3 allora H_1 è simile ad H_3.

Il metodo QR venne pubblicato indipendemente nel 1961 da Francis e da Kublanovskaya 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 esistono vari risultati (cf. [7], p. 393, [13, p.180], [11, p.21, Thm. 5.3]). Ricordiamo principalmente [6, p.149]

Teorema 5.1 *Se la matrice $A\mathbb{R}^{n\times n}$ è qualsiasi ma regolare e con autovalori tutti distinti in modulo, con*
$$|\lambda_1| > \text{ÂůÂůÂů} > |\lambda_n|$$
, allora lâĂŹalgoritmo QR converge ad una matrice A_∞ triangolare superiore. Se la matrice è simmetrica, allora
$$A_\infty = diag(\lambda_1, \ldots, \lambda_1).$$
Inoltre se A è una matrice Hessenberg superiore allora lâĂŹalgoritmo QR converge e la matrice A_∞ triangolare a blocchi, simile ad A e gli autovalori di ogni blocco diagonale sono tutti uguali in modulo.

Alcuni dettagli.

1. 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. In entrambi i casi le iterazioni mantengono la struttura, cioè se $A_0 = T$ è di Hessenberg, allora A_k è di Hessenberg, se $A_0 = T$ è tridiagonale allora A_k è tridiagonale.

2. Per inciso, si può dimostrare che la velocità di convergenza dipende dal rapporto

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

Il numero di moltiplicazioni necessarie allâĂŹalgoritmo di Givens per calcolare tale matrice T a partire da A è approssimativamente $10n^3/3$ mentre per quanto riguarda l'algoritmo di Householder è $5n^3/3$ [14, p.252]. Il metodo QR applicato ad una matrice A in forma di Hessenberg superiore ha ad ogni passo un costo di $2n^2$ operazioni moltiplicative.

3. Se la condizione (**??**) 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 [7], p. 394.

5.1 Facoltativo: Metodo QR con shift

la velocità di convergenza dipende dal rapporto ρ in (26). 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 $\{\lambda_i - \mu\}_{i=1,\ldots,n}$ da cui si ottengono facilmente $\{\lambda_i\}_{i=1,\ldots,n}$.
Esistono diverse varianti di questa tecnica.

5.1.1 Variante 1

In una prima (cf. [3], p. 363) si pone

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

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 $(a_k)_{n,n}$.

5.1.2 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.

6 Esperimenti numerici in Matlab

6.1 Il metodo delle potenze

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
```

Qualche nota

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. [12, p.180])

6.2 Esempio 1

Testiamo il codice relativamente al calcolo dell'autovalore di massimo modulo di

$$\begin{aligned} A &= \begin{pmatrix} -15.5 & 7.5 & 1.5 \\ -51 & 25 & 3 \\ -25.5 & 7.5 & 11.5 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix} \cdot \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix}^{-1} \end{aligned} \tag{23}$$

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 $\alpha_1 = 7/6 \neq 0$. 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;
>> [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
  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-004
  2.0924e-005
  2.0924e-006
  2.0924e-007
  2.0924e-008
  2.0924e-009
>>
```

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
```

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
>> [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
>>
```

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.

6.3 Esempio 2

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 $\lambda_1 \neq \lambda_2$. 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
 -3.4483e-002
 -2.0000e-001
 -3.4483e-002
 -2.0000e-001
 -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
  4.4567e-001
  2.4000e+000
  4.4567e-001
  2.4000e+000
  4.4567e-001
  2.4000e+000
  4.4567e-001
  2.4000e+000
  4.4567e-001
  2.4000e+000
>>
```

Dal residuo pesato è chiaro che il metodo non converge, e come anticipato il motivo è la presenza di autovalori distinti aventi modulo massimo.

6.4 Esempio 3

Per terminare, vediamo il caso della matrice diagonalizzabile (avendo autovalori distinti)

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

in cui il metodo funziona rapidamente, in quanto esiste un solo autovalore di modulo massimo, uguale a 10.

```
>> A=[1 2; 0 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
  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
  9.7618e-005
  9.7619e-006
  9.7619e-007
  9.7619e-008
  9.7619e-009
```

Si osservi che il metodo termina in quanto l'errore pesato `err` è minore della tolleranza $toll = 10^{-8}$.

6.5 Il metodo delle potenze inverse

Una versione di base `invpower` del metodo delle potenze inverse [12, 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 16, 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.

6.6 Esempio 1

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

$$\begin{aligned} A &= \begin{pmatrix} -15.5 & 7.5 & 1.5 \\ -51 & 25 & 3 \\ -25.5 & 7.5 & 11.5 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix} \cdot \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 9 & 3 \end{pmatrix}^{-1} \end{aligned} \tag{24}$$

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 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 inverse applicato ad A, e avente quale punto iniziale z_0 può essere utilizzato per il calcolo dell'autovalore di minimo modulo di A.

```
>> 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]
A =
 -15.50000000000000   7.50000000000000   1.50000000000000
 -51.00000000000000  25.00000000000000   3.00000000000000
 -25.50000000000000   7.50000000000000  11.50000000000000
>> [lambda, x, niter, err]=invpower(A,z0,mu,toll,nmax)
lambda =
   0.39016115351993
   0.94237563941268
   0.99426922936854
   0.99942723776656
   0.99994272692315
   0.99999427272378
```

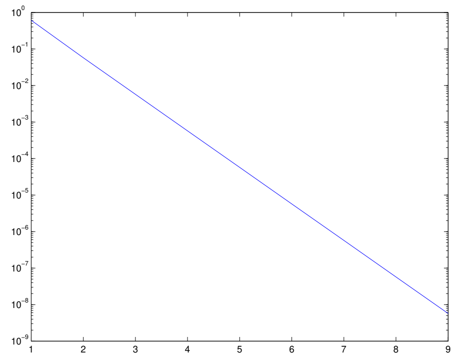

Figura 2: Grafico che illustra la convergenza lineare del metodo delle potenze inverse nell'esempio 1.

```
   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
>>
```

La convergenza è lineare (come si intuisce dalle approssimazioni contenute nel vettore `lambda`).

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.

6.7 Facoltativo: Esempio 2: matrice di Poisson

Vogliamo ora applicare il metodo delle potenze e delle potenze inverse per il calcolo di alcuni autovalori di una matrice di Poisson

$$A = \begin{pmatrix} B & -I & 0 & \dots & 0 \\ -I & B & -I & \dots & 0 \\ 0 & -I & B & \dots & \dots \\ 0 & \dots & \dots & \dots & -I \\ 0 & 0 & \dots & -I & B \end{pmatrix}$$

con

$$B = \begin{pmatrix} 4 & -1 & 0 & \dots & 0 \\ -1 & 4 & -1 & \dots & 0 \\ 0 & -1 & 4 & \dots & \dots \\ 0 & \dots & \dots & \dots & -1 \\ 0 & 0 & \dots & -1 & 4 \end{pmatrix}.$$

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.

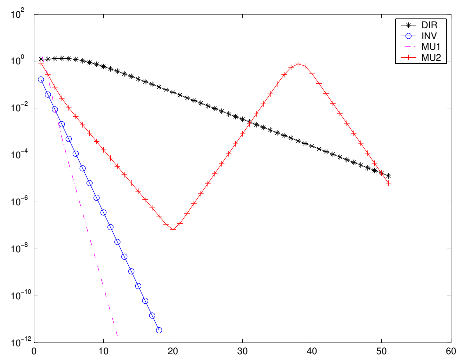

Figura 3: Grafico che illustra il residuo per iterazione del metodo delle potenze diretto e nelle versioni di Wielandt.

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

          [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.

6.8 Il metodo QR

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
```

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
```

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
```

7 Sugli autovalori della matrice di Poisson

In questa sezione desideriamo calcolare gli autovalori della matrice (simmetrica e a banda) 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.

8 Esercizio

Si può dimostrare (cf. [18]) che gli zeri del polinomio monico di grado n

$$p(t) = c_0 + c_1 t + \cdots + c_{n-1} t^{n-1} + t^n$$

sono gli autovalori della matrice compagna (o equivalentemente della sua trasposta visto che $eig(A) = eig(A^T)$)

$$C = \begin{bmatrix} 0 & 0 & \dots & 0 & -c_0 \\ 1 & 0 & \dots & 0 & -c_1 \\ 0 & 1 & \dots & 0 & -c_2 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -c_{n-1} \end{bmatrix}.$$

Sfruttando il comando `eig`, e osservando che

```
>> A=[1 2; 3 4]

A =

     1     2
     3     4

>> B=[5 6; 7 8]

B =

     5     6
     7     8

>> C=[A B]

C =

     1     2     5     6
     3     4     7     8

>> D=[A; B]

D =

     1     2
     3     4
     5     6
     7     8

>>
```

scrivere un codice Matlab che risolva un'equazione algebrica calcolando gli autovalori della matrice compagna associata. Effettuare quindi un test su un polinomio di cui si conoscono gli zeri e valutare la bontà del proprio codice. Di seguito si calcolino gli zeri di modulo massimo e minimo usando il metodo delle potenze e la sua variante inversa.

9 Facoltativo: Alcuni esempi ed esercizi

1. Lanciare il codice `demoQR` relativamente alla matrice `makefish(3)` (e non `makefish(2)`). A tal proposito modificare la variabile `siz`.

2. 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.

3. 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. [7], p. 393). Suggerimento: ricordarsi come calcolare gli autovalori di una matrice di ordine 2 via equazioni di secondo grado.

4. 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. [12], p. 195).

5. 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. [13], p.133).

6. 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. [13], p. 141, esercizio 6.5).

10 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. [7], 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. [2], p. 625)

Teorema 10.1 *Sia A una matrice di ordine n e supponiamo che i suoi autovalori $\{\lambda_i\}_{i=1,\ldots,n}$ soddisfino*

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

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

$$\|D - A_m\| \leq c \max_i \frac{|\lambda_{i+1}|}{|\lambda_i|} \tag{25}$$

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

$$\rho = \max_i \frac{|\lambda_{i+1}|}{|\lambda_i|}. \tag{26}$$

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

11 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] E. Andersson e P.A. Ekström, *Investigating Google's PageRank algorithm*, http://www.it.uu.se/edu/course/homepage/projektTDB/vt04/projekt5/website/report.pdf.

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

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

[4] C. Brezinski e M. Redivo Zaglia, *Some numerical analysis problems behind web search*, Transgressive Computing 2006.

[5] C. Brezinski e M. Redivo Zaglia, *The PageRank vector: properties, computation, approximation and acceleration*, SIAM J. Matr. Anal. Appl., Vol.28, N.2, p. 551-575.

[6] C. Brezinski e M. Redivo Zaglia, *Méthodes numériques itératives*, Ellipses, 2006.

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

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

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

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

[11] P. Olver, *Notes on orthogonal bases and the workings of the QR algorithm* http://www.math.umn.edu/ olver/aims_/qr.pdf

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

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

[14] M. Redivo Zaglia, *Calcolo numerico. Metodi e algoritmi*, quarta edizione., L. Progetto, 2009.

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

[16] Wikipedia (Eigenvalue) http://en.wikipedia.org/wiki/Eigenvalue.

[17] Wikipedia (Inverse iteration) http://en.wikipedia.org/wiki/Inverse_iteration.

[18] Wikipedia (Matrice Compagna)
http://it.wikipedia.org/wiki/Matrice_compagna.

[19] Wikipedia (Metodo delle potenze)
http://it.wikipedia.org/wiki/Metodo_delle_potenze.

[20] Wikipedia (PageRank)
http://it.wikipedia.org/wiki/Page_rank.

[21] Wikipedia (QR algorithm)
http://en.wikipedia.org/wiki/QR_algorithm.

[22] Wikipedia (QR decomposition)
http://en.wikipedia.org/wiki/QR_decomposition.

[23] Wikipedia (Teoremi di Gerschgorin)
http://it.wikipedia.org/wiki/Teoremi_di_Gerschgorin.