Accelerazione funzionale al calcolo dei vettori dell' HITS ExpertRank

Gini Agnese

5 marzo 2014

Sommario

Un'analisi e rielaborazione dell'articolo *Practical Acceleration for Computing the HITS ExpertRank Vectors* di *Yunkai Zhou* del 2011, il quale espone una metodo basato sul filtro di Chebyshev applicato al metodo delle potenze per rendere il calcolo degli autovettori del ranking fino a due volte più veloce, rispetto al metodo standard.

1 Introduzione

Il problema della ricerca di autovettori per matrici non negative ha trovato una delle principali coniugazioni nel *ranking* delle pagine web. Le principali applicazioni in questo campo sono infatti il PageRank di Google e l'Hyperlink-Induced Topic Search (HITS).
L'HITS fu sviluppato da Kleinberg negli anni novanta ed è usato come motore di ricerca da `Ask.com`. Il due vettori che sono forniti dall'implementazione dell'HITS danno i così detti ExpertRanks. Nel seguito verranno discussi sia il modello di HITS originale sia le sue modifiche.
Come noto il problema sostanziale nella classificazione della pagine web è l'ordine di grandezza del numero degli oggetti coinvolti. Il principale metodo adottato per la risoluzione del PageRank e dell'HITS, a causa delle limitazioni di memoria, è il *metodo delle potenze*. È noto che se il rapporto tra i valori assoluti dei primi due autovalori, in ordine di grandezza, (*gap-ratio*) è vicino ad 1 il metodo può essere inefficiente, ma è anche vero che tendenzialemente per il PageRank questo è circa 0.85 e dunque il metodo ha una buona convergenza. Sono stati proposti molti accorgimenti per la velocizzazione degli algoritmi per il PageRank, si trovano invece pochi lavori riguardo l'HITS. Si può ricondurre ciò al fatto che in primo luogo che la grandezza delle matrici coinvolte spesso non è enorme, in secondo luogo perché il rapporto degli autovalori non è prossimo a 1.
Verrà nel seguito proposto uno schema di accelerazione basato sul *metodo delle potenze filrato*, ciò implica inoltre che esso potrà trovare applicazioni che vanno oltre l'HITS basilare.
È importante osservare che ciò preso in analisi non si applica al PageRank, è infatti necessario che tutti gli autovalori siano reali.

2 Modello HITS

Il modello HITS descrive la struttura dei collegamenti ipertestuali tra le pagine tramite un grafo orientato. Le pagine sono suddivise in due categorie: Hubs e Authorities. L'HITS parte dal fatto che un buon ordinamento è ottenibile tramite mutue applicazioni tra le due classificazioni: infatti validi Hubs tendenzialemente puntano a buone Authorities, buone Authorities sono generalmente puntati da validi Hubs.
Il modello HITS calcola dunque due vettori. Ci si aspetta che le pagine con un alto grado d'Authority abbiano un contenuto attinente, mentre quelle con un alto grado d'Hub contengano dei collegamenti a contenuti attinenti.

Denotiamo $\mathbf{v}_\mathrm{h}$ e $\mathbf{v}_\mathrm{a}$ rispettivamente l'Hub rank e l'Authority rank. Sia poi L la matrice di adiacenza del grafo orientato, $L \in \mathcal{M}(n)$. Si può dunque utilizzare, a partire dai dati iniziali $\mathbf{v}_\mathrm{h}^{(0)}$ $\mathbf{v}_\mathrm{a}^{(0)}$, una procedura iterativa per trovare $\mathbf{v}_\mathrm{h}$ e $\mathbf{v}_\mathrm{a}$

$$\mathbf{v}_\mathrm{a}^{(\mathrm{k})} = L^T \mathbf{v}_\mathrm{h}^{(\mathrm{k}-1)} \ , \ \mathbf{v}_\mathrm{h}^{(\mathrm{k})} = L \mathbf{v}_\mathrm{a}^{(\mathrm{k}-1)} \ , \ k = 1, 2, 3... \tag{1}$$

E combinando le due equazioni si ottiene

$$\mathbf{v}_\mathrm{a}^{(\mathrm{k})} = L^T L \mathbf{v}_\mathrm{a}^{(\mathrm{k}-1)} \ , \ \mathbf{v}_\mathrm{h}^{(\mathrm{k})} = L L^T \mathbf{v}_\mathrm{h}^{(\mathrm{k}-1)} \tag{2}$$

Le equazioni (2) danno il metodo delle potenze, senza normalizzazione, applicato a LL^T e $L^T L$; normalizzando si ottengono comunque gli autovettori unitari principali delle due matrici in questione:

$$LL^T \mathbf{v}_\mathrm{h} = \lambda_{max} \mathbf{v}_\mathrm{h} \tag{3}$$

$$L^T L \mathbf{v}_\mathrm{a} = \lambda_{max} \mathbf{v}_\mathrm{a} \tag{4}$$

con λ_{max} l'autovalore di modulo massimo di entrambe.

Al fine di garantire la convergenza si assume che λ_{max} sia unico. Nel caso in cui questa ipotesi non si verifichi si può utilizzare il metodo delle potenze applicato alla seguente matrice, per cui invece è unico,

$$\xi L^T L + \frac{1-\xi}{n} \mathbf{e}\mathbf{e}^\top, \ \ \xi L L^T + \frac{1-\xi}{n} \mathbf{e}\mathbf{e}^\top, \ dove \ \mathbf{e} = [1, 1, \ldots, 1]^T \tag{5}$$

Si osserva che il gap-ratio di LL^T e $L^T L$ nel modello HITS è ragionevolmente piccolo, il che induce che la convergenza sia ragionevolmente veloce.
Tuttavia, non ci sono risultati teorici che garantiscano che il gap-ratio sia ridotto. Vi sono in effetti numerosi esempi in cui si constata questa effettiva problematica. Sarebbe dunque auspicabile riuscire ad accelerare il metodo delle potenze a dispetto del gap-ratio.

3 Algoritmo principale

Partendo dall'osservare che $\mathbf{v}_\mathrm{a} = L^T \mathbf{v}_\mathrm{h}$, la trattazione si concentrerà principalmente sulla ricerca dell'Hub rank (equivalentemente può essere fatta a partire dall'Authority rank).
Le matrici LL^T e $L^T L$ sono simmetriche, definite positive e con spettro reale. Idealmente queste proprietà si adattano bene col proposito di usare filtri di Chebyshev per accelerare il calcolo dei vettori dell'ExpertRank.
L'algoritmo proposto si basa su tali filtri per il metodo delle potenze, usando un solo vettore in più rispetto al metodo delle potenze classico, la ricorrenza dei polinomi di Chebyshev è infatti a tre termini. Si ha pertanto che l'uso della memoria è comparabile a quello del metodo delle potenze (e più economico rispetto a quello dei sottospazi di Krylov [1]). Dal punto di vista computazionale, questo sistema necessita in più di un solo prodotto per calcolare un quoziente di Rayleigh[2] per m prodotti matrice-vettore, con m grado del polinomio. Si ha quindi che per un utilizzo dei polinomi di Chebyshev mirato al filtraggio, è necessario determinare gli estremi da usare. Esperienze numeriche mostrano che l'utilizzo standard non porta a buoni risultati, tuttavia con opportuni accorgimenti si ottengono dei risultati accettabili.

[1] Vedi 6.1 sottospazio di Krylov
[2] Vedi 6.2 quoziente di Rayleigh

3.1 Stima della banda per il filtraggio

È noto che i polinomi di Chebyshev sono limitati in [-1,1] e hanno un incremento esponenziale fuori da tale intervallo. Si rende dunque necessario mappare linearmente la parte di spettro indesiderata in [-1,1]. In questo modo si ha anche che automaticamente la parte voluta è mappata fuori, perciò aumentata dai filtri polinomiali di Chebyshev.
Per l'applicazione all'ExpertRank si vuole calcolare l'autovettore associato all'autovalore di modulo massimo, dunque il filtro dovrà smorzare la coda dello spettro e dato che gli spettri di LL^T e L^TL sono reali risulta conveniente cercare di portarla a 0.
È comunque quello che verrà chiamato *filtering upper bound*, ossia l'estremo superiore dello spettro indesiderato, che determina l'effettiva efficienza del filtro. Questo dovrà essere infatti più piccolo del più grande autovalore di LL^T, denotato $\lambda_{max}(LL^T)$, così che $\lambda_{max}(LL^T)$ sia mappato fuori da [-1,1], cosicché venga opportunamente ingrandito e non diminuito dal filtro. Per sistemare gli estremi sono sfruttabili le proprietà variazionali date dal teorema min-max di Courant-Fisher [3]. E ciò verrà fatto in due passi.

Come primo passo cerchiamo due valori $0 < u_l < u_L$ tramite la procedura di Lanczos. Si osserva che essendo la l'estremo superiore da stimare e la matrice in esame definita positiva si ha che quello che veramente interessa è u_l ($u_l < \lambda_{max}(LL^T)$). u_L è meno essenziale, è solo una stima approssimativa di $\lambda_{max}(LL^T)$. L'Algoritmo 3.1 mostra la procedura per individuare questi due valori. In particolare si nota che viene fatto uso del passo k-esimo della procedura per il calcolo della decomposizione standard di Lanczos[4]

$$LL^TV_k = V_kT_k + f_ke_k^T,\ V_k^TV_k = I_k \text{ e } V_k^Tf_k = \mathbf{0}$$

con V_k matrice che contiene i k vettori ortonormali di Lanczos. L'autovalore esterno di LL^T può essere stimato approssimativamente dal quoziente di Rayleigh della tridiagonale T_k.
E a partire dal fatto che è necessario solo un'approssimazione di $\lambda_{max}(LL^T)$, si può usare k piccolo, ad esempio k=3, per ridurre la quantità di memoria impiegata. E essendo $u_L \geq \lambda_{max}(LL^T)$ non necessaria si può usare la valutazione di Ritz [5] più grande (più un termine d'errore portato dalla decomposizione). Il vettore v_0, infine, può essere un qualsiasi vettore nonnullo.

Algoritmo 3.1. .

```
function [ul, uL,v0]=Lanczos_bounds(k, v0, L)
%costrizione decomposizione parziale di Lanczos
V(:,1)=v0/(norm(v0,2));
V(:,2)=L*(L'*V(:,1));
a=V(:,2)'*V(:,1);
V(:,2)=V(:,2)-a*V(:,1);
T(1,1)=a;
m=min(k,5);
for j=2:m,
    h=norm(V(:,j));
    V(:,j)=V(:,j)/h;
    V(:,j+1)=L*(L'*V(:,j));
    V(:,j+1)=V(:,j+1)+h*V(:,j-1);
    a=V(:,j+1)'*V(:,j);
    V(:,j+1)=V(:,j+1)-a*V(:,j);
    T(j,j-1)=h;
    T(j-1,j)=h;
```

[3] Vedi 6.2 Teorema minmax di Courant-Fisher
[4] Decomposizione di Lanczos 6.3
[5] Coppia di Ritz 6.3

```
18      T(j,j)=a;
19  end
20  [Q,D]=eig(T(1:m,1:m));      %ricerca massimo autovettore per T
21  d=diag(D);
22  [uL,idx]=max(d);          %aggiornamento valori stima
23  ul=(min(d) + uL)/2;
24  uL=uL+h*Q(end,idx);
25  v0=V(:,1:m)*Q(:,idx);
26  s=sign(v0);
27  v0=s(1)*v0;
28  end
```

All'ultimo passo dell'algoritmo si pone $u_l = (\lambda_{min}(T_k) + \lambda_{max}(T_k))/2$; in questo ottiamo le relazioni $u_l \leq \lambda_{max}(T_k) < \lambda_{max}(LL^T)$, in accordo con le proprietà di un quoziente di Rayleigh.
Si ha anche un altro vantaggio, l'algoritmo infatti fornisce il vettore di Ritz corrispondente al valore di Ritz più grande dato dal k-esimo passo di Lanczos, che in generale è migliore del vettore ottenuto col metodo delle potenze allo stesso passo, e tale vettore è migliore da usare come vettore iniziale per il metodo delle potenze col filtro di Chebyshev. Si nota inotre che il valore u_L è utilizzabile eventualmente per riscalare, ma si vedrà che è inutile per il proseguo.

Il secondo passo, che contiene gli stadi fondamentali del metodo delle potenze col filtro di Chebyshev, semplicemte migliora u_l ad ogni iterazione. Con tale scopo, operiamo una combinazione convessa del valore al passo precedente e un quoziente di Rayleigh u_u calcolato agilmente tramite il filtro di Chebyshev (Algoritmi 3.2 e 3.3).

$$u_l \leftarrow \beta u_l + (1-\beta)u_u, \text{ con } \beta \in (0,1) \tag{6}$$

Per il teorema di Courant-Fisher si ha $u_u < \lambda_{max}(LL^T)$ prima della convergenza. E ciò, unito al fatto che ad ogni passo $u_l < \lambda_{max}(LL^T)$ da che u_l fornito da (6) sarà sempre più piccolo dell'autovalore maggiore di LL^T.
Mappando dunque $[0,u_l]$ in [-1,1] è possibile garantire che il filtro smorzerà la parte di spettro indesiderata e amplificherà la regione in cui si trova $\lambda_{max}(LL^T)$ incrementando il gap-ratio e di conseguenza la velocità di convergenza del metodo delle potenze.

3.2 Algoritmo HITS con accelerazione di Chebyshev

Il ben noto polinomio di grado k di Chebyshev di prima specie è definito

$$C_k(t) := \begin{cases} \cos(k\ \cos^{-1}(t)) & |t| \leq 1 \\ \cosh(k\ \cosh^{-1}(t)) & t > 1 \\ (-1)^k \cos(k\ \cosh^{-1}(-t)) & t < -1 \end{cases} \tag{7}$$

Che si traduce in una ricorrenza a tre termini

$$\begin{cases} C_{k+1}(t) = 2tC_k(t) - C_{k-1}(t) \ \ t \in \mathbb{R},\ k \in \mathbb{N}^+ \\ C_0(t) = 1 \\ C_1(t) = t \end{cases} \tag{8}$$

Dalla scrittura in (7) si possono notare le proprietà di convergenza nei vari intervalli sopracitate.

Proposizione. *Dato il polinomio di Chebyshev di prima specie $C_k(t)$ vale che:*

- $|C_k(t)| < 1$ *per* $t \in [-1,1]$
- *ha crescita esponenziale per* $t \in \mathbb{R} \setminus [-1,1]$

Rimane da esplicitare la mappa che porta [0, u_l] in [-1,1]: $\mathcal{L}(t) = (t - \frac{u_l}{2})(2/u_l)$. Questa applicazione affine applicata ad una qualsiasi matrice con autovalori in [0, u_l] in una $\mathcal{L}(A)$ con autovalori in [-1,1], con prodotto vettore associato

$$\mathcal{L}(A)v = (Av - \frac{u_l}{2}v)(2/u_l).$$

Algoritmo 3.2. .

```
function [Y,uu]=Chebyshev_filter(X,m,ul,L)
%Mappa affine
e=ul/2;
Y=(L*(L'*X))/e-X;
%calcolo del polinomi fino al grado m-1
for i=1:(m-1)
    Z=(L*(L'*Y)-e*Y)*(2/e) - X;
    X=Y;
    Y=Z;
end
Z=L*(L'*Y);
uu=(Z'*Y)/(Y'*Y);   %quoziente di Rayleigh
Y=(Z-e*Y)*(2/e)-X;  %polinomio di grado m
end
```

Dato un vettore X in input l'algoritmo essenzialmente calcola il vettore filtrato $Y = C_m(\mathcal{L}(LL^T))X$, e per tale calcolo si usa la ricorrenza a tre termini (8). L'output u_u è il quoziente di Rayleigh corrente che sarà usato per il riscalo al passo successivo.
La versione dell'algoritmo scalato può essere utile per prevenire casi overflow quado il grado m è grande e non ci sono altri riscali.

Algoritmo 3.3. .

```
function [Y,uu]=Chebyshev_filter_scaled(X,m,ul,uL,L)
%Mappa affine
e=ul/2;
s=e/(uL-e);
t=2/s;
Y=(L*((L')*X)-e*X)*(s/e);
%calcolo del polinomio fino al grado m-1
for i=1:(m-1)
    sn=1/(t-s);
    Z=(L*((L')*Y)-e*Y)*(2*sn/e) -(s*sn)*X;
    X=Y;
    Y=Z;
    s=sn;
end
sn=1/(t-s);
Z=L*((L')*Y);
uu=((Z')*Y)/(Y'*Y);      %quoziente di Rayleigh
Y=(Z-e*Y)*(2*sn/e)-(s*sn)*X;     %polinomio di grado m
end
```

Lo scopo di questo algoritmo è ancora una volta filtrare il vettore un input X con un polinomio di Chebyshev che riproporzioni l'intervallo [0,u_l], ma qui si applica $C_m(\frac{1}{e}(u_L - e))$ come fattore si scalo. Si nota che i valori in imput debbono rispettare le disuguaglianze precedentemente discusse.

L'assegnamento $e = u_l/2$ determina l'unica variabile per la trasformazione affine, le altre due `s` e `t` sono usate per l'aggiornamento del polinomio.

Sostanzialmente l'algoritmo calcola

$$Y = \frac{C_m(\frac{1}{e}(LL^T - eI))}{C_m(\frac{1}{e}(u_L - e))}X \tag{9}$$

Il filtro rinormalizzato dunque non solo rimpicciolisce $[0, u_l]$, ma simultaneamente aumenta gli autovalori in $[u_l, \lambda_{max}(LL^T)]$.
Questo per la continuità dei polinomi di Chebyshev e le proprietà di crescita.
L'Algoritmo 3.3 è leggermente più conveniente in termini di costi anche se prevede il convolgimento di più dati iniziali.

Si ha infine che l'Algoritmo 3.4 descrive l'algoritmo per il calcolo del vettore Hub dell'ExpertRank. Questo si basa principalmente sul prodotto matrice vettore $L(L^T)v$.

Un vettore di ranking è un autovettore principale della matrice non negativa LL^T, riscalando uno vettore di questo tipo la classificazione non cambia. Di solito si usa però un vettore a componeti non negative.
È importante osservare che il questo caso specifico la matrice di adiacenza è verosimilmente riducibile essendo il grafo del web non fortemente connesso. Dunque non si può applicare il Teorema di Perron-Frobenius[6] che garantirebbe l'esistenza di un autovalori a componenti positive.
Il vettore di input X_0 dovrà essere positivo, l'eventualità per cui compaiano entrate negative per il filtro è compensata ai passi 21-22.

Algoritmo 3.4. .

```
1  function [X,it]=HITS_Chebyshev(X0, m, b, f, itmax, method, L)
2  tic
3  k=5;            % # iterazioni lanczos
4  [ul, uL, X0]=Lanczos_bounds(k, X0, L);
5  uu=uL;
6  it=0;
7  X0=X0/(norm(X0,1)); %normalizzazione dato iniziale
8  for i=1:itmax       %iterazioni metodo
9      if (method=='scaled')
10         uL=max(uu,uL);
11         [X,uu]=Chebyshev_filter_scaled(X0,m,ul,uL,L);
12     else
13         [X,uu]=Chebyshev_filter(X0,m,ul,L);
14     end
15     %normalizzazione vettore attuale
16     o=sign(X);
17     X=o(1)*X/(norm(X,1));
18     %valutazione del passo per condizione di arresto
19     if (norm((X-X0),1)<f) break
20     end
21     X0=X;
22     ul=b*ul+(1-b)*uu;
23 it=it+1;
24 end
25 if (norm((X-X0),1)>f) disp('itmax') %controllo
26 end
27 toc
```

[6]Vedi Teorema di Perron-Frobenius 6.3

```
28  end
```

In conclusione si ha un algoritmo più complesso del metodo delle potenze. Ognuno degli m prodotti matrice-vettore di $(LL^T)^m X_0$ è sostituito da $C_m(\mathcal{L}(LL^T))X_0$.Il numero di moltiplicazioni matrice vettore per passo è

- 6 metodo delle potenze **classico**
- 12 metodo delle potenze **filtrato**

Dal punto di vista della memoria invece c'è solo la memorizzazione di un solo vettore in più, per iterazione di Lanczos, rispetto al metodo delle potenze.

4 Derivazione e analisi algoritmi

Gli Algoritmi 3.2 e 3.3 costruiscono un filtro che smorza l'intervallo $[0, u_l]$, in generale la mappa affine che porta un intervallo [a,b] in [-1,1] è

$$\mathcal{L}(t) = \frac{t-c}{e}, \text{ dove } c = \frac{b-a}{2} \text{ e } e = \frac{b+a}{2}.$$

Un intervallo è univocamente determinato dal centro c e matà ampiezza e.

Si consideri ora la ricorrenza a tre termini di Chebyshev (7) applicata a $\mathcal{L}(t)$:

$$C_{k+1}(\mathcal{L}(t)) = 2\mathcal{L}(t)C_k(\mathcal{L}(t)) - C_{k-1}(\mathcal{L}(t)),\ t \in \mathbb{R},\ k = 1, 2, \dots \tag{10}$$

con $C_0(t) \equiv 1$ e $C_1(\mathcal{L}(t)) = \mathcal{L}(t)$.

Definiamo $x_k = C_k(\mathcal{L}(A))x_0$ per un qualsiasi vettore condizione iniziale (notare che $x_1 = \frac{1}{e}(A - cI)x_0$) ma allora (10) porta alla ricorrenza

$$x_{k+1} = C_{k+1}(\mathcal{L}(A))x_0 = \frac{2}{e}(A - cI)x_k - x_{k-1},\ k = 1, 2, \dots\ . \tag{11}$$

L'iterazione può essere scritta anche come

$$X_k = \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -I & \frac{2}{e}(A - cI) \end{bmatrix} \begin{bmatrix} x_{k-1} \\ x_k \end{bmatrix} = \mathcal{B}X_{k-1} \tag{12}$$

L'equazione (12) mostra che essenzialmente la (11) non è altro che l'iterazione della potenza della matrice $\mathcal{B}$, e dunque la convergenza può essere studiata in termini degli autovalori di tale matrice.

Lemma 4.1. *Sia $M \in \mathcal{M}(n)$ con autovalori d_i con $i = 1, \dots, n$ $\Rightarrow$ gli autovalori della matrice $\mathcal{M}(2n) \ni \begin{bmatrix} 0 & I \\ -I & M \end{bmatrix}$ sono $\frac{d_i \pm \sqrt{d_i^2 - 4}}{2}$ $i = 1, \dots, n$.*

Dimostrazione. La matrice ha rango massimo, quindi ha tutti autovalori $\mu \neq 0$. Usando le proprietà del determinante $det \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} = det(D_{11})det(D_{22} - D_{21}D_{11}^{-1}D_{12})$ con D_{ij} quadrate.

Allora $det \begin{bmatrix} -\mu I & I \\ -I & M - \mu I \end{bmatrix} = det(-\mu M + \mu^2 I + I) = \prod_{i=1}^{n}(\mu^2 - d_i\mu + 1)$. E dunque gli autovalori sono le radici dei fattori, ossia $\frac{d_i \pm \sqrt{d_i^2 - 4}}{2}$ $i = 1, \dots, n$.

✓

Denotiamo dunque gli autovalori di A come $\lambda_i(A)$. Possiamo applicare il Lemma 4.1 con $M = \frac{2}{e}(A - cI)$ e si ha quindi

$$\lambda_i^{(1,2)}(\mathcal{B}) = \frac{d_i \pm \sqrt{d_i^2 - 4}}{2}, \text{ dove } d_i = \frac{2}{e}(\lambda_i(A) - c) \; i = 1, \ldots, n. \tag{13}$$

Osservazione. *Si che*

- *se $\lambda_i(A) \in [a, b]$, allora $\frac{2}{e}(\lambda_i(A) - c) \leq 2$ e i corrispondenti $\lambda_i^{(1,2)}(\mathcal{B})$ sono complessi coniugati e di modulo 1;*
- *se $\lambda_i(A) \in \mathbb{R} \setminus [a, b]$, i corrispondenti $\lambda_i^{(1,2)}(\mathcal{B})$ sono reali e quello norma maggiore è fuori dall'intervallo [-1,1].*

Il che ci porta alla convergenza dell'iterazione di Chebyshev.

Lemma 4.2. *Se l'autovalore massimo di una matrice hermitiana A unico e l'inervallo di filtraggio soddisfa $a \leq \min_{i=1,\ldots,n} \lambda_i(A)$ e $b \leq \max_{i=1,\ldots,n} \lambda_i(A) \Rightarrow$ il termine k-esimo della (11) converge all'autovettore principale di A.*

Dimostrazione. Ordiniamo gli autovalori in ordine non crescente $\lambda_1(A) > \lambda_2(A) \geq \cdots \geq \lambda_n(A)$, con d_i definiti come in (13). Essendo $a \leq \lambda_n(A)$ e $b < \lambda_1(A)$ si ha che $\max_{i=1,\ldots,n} \lambda_i^{(1,2)}(\mathcal{B}) = \frac{d_1 + \sqrt{d_1^2 - 4}}{2} > 1$, unico autovalore principale di $\mathcal{B}$ Che ci da, visto nell'ottica dell'iterazione (12), la convergenza di x_k in (11). Ma x_k converge all'autovettore di principale di A poichè $x_k = C_k(\mathcal{L}(A))x_0 = C_k(\frac{1}{e}(A - cI))x_0$, e scalare e traslare una matrice non modifica gli autovettori. ✓

L'iterazione (11) può essere ulteriormente scalata in favore di una maggiore stabilità. Una semplice strategia è quella di sostituire $C_k(\mathcal{L}(A))$ nel calcolo $x_k = C_k(\mathcal{L}(A))x_0$ con

$$\tilde{C}_k(\mathcal{L}(A)) := \frac{C_k(\frac{1}{e}(A - cI))}{\rho_k}, \text{ dove } \rho_k := C_k(\frac{1}{e}(\tilde{b} - c)). \tag{14}$$

Dove $\tilde{b}$ è un valore fuori da [a,b]. In questo caso $|\frac{1}{e}(\tilde{b} - c)| > 1$ e il fattore di riscalo aumenta esponenzialmente con k migliorando la stabilità numerica. Tuttavia se diventasse troppo grande l'applicazione diventerebbe insignificante, che è controproduttivo per l'accelerazione. Dunque $|\tilde{b}|$ non deve essere troppo grande di $\lambda_{max}(LL^T)$.
Con alcune osservazioni si ottiene una ricorrenza a tre termini anche in questo caso:

$$\sigma_k = \rho_k / \rho_{k+1}$$

$$\tilde{C}_{k+1}(\mathcal{L}(A)) = \sigma_{k+1}\frac{2}{e}(A - cI)\tilde{C}_k(\mathcal{L}(A)) - \sigma_{k+1}\sigma_k\tilde{C}_{k-1}(\mathcal{L}(A)), \;\; k = 1, 2, \ldots \; . \tag{15}$$

e

$$\sigma_1 = \frac{\rho_0}{\rho 1} = \frac{e}{\tilde{b} - c}, \; \sigma_{k+1} = \frac{C_k(\frac{1}{e}(\tilde{b} - c)}{C_{k+1}(\frac{1}{e}(\tilde{b} - c)} = \frac{1}{2/\sigma_1 - \sigma_k}. \tag{16}$$

Presa $\tilde{x}_0$ come condizione iniziale si ottiene $\tilde{x}_1 = C_1(\mathcal{L}(A))\tilde{x}_0 = \frac{\sigma_1}{e}(A - cI)\tilde{x}_0$ e applicando (15) e (16)

$$\tilde{x}_{k+1} = 2\frac{\sigma_{k+1}}{e}(A - cI)\tilde{x}_k - \sigma_{k+1}\sigma_k\tilde{x}_{k-1}, \;\; k = 1, 2, \ldots \; . \tag{17}$$

Che come per il caso precedente può essere espressa in termini matriciali

$$\tilde{X}_k = \begin{bmatrix} \tilde{x}_k \\ \tilde{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -\sigma_{k+1}\sigma_k I & \frac{2\sigma_{k+1}}{e}(A - cI) \end{bmatrix} \begin{bmatrix} \tilde{x}_{k-1} \\ \tilde{x}_k \end{bmatrix} = \tilde{\mathcal{B}}\tilde{X}_{k-1} \tag{18}$$

E con argomentazioni analoghe a quelle del caso trattato prima si ottiene

Lemma 4.3. *Se l'autovalore massimo di una matrice hermitiana A è unico, l'inervallo di filtraggio soddisfa*

$$a \leq \min_{i=1,\dots,n} \lambda_i(A) \; e \; b \leq \max_{i=1,\dots,n} \lambda_i(A),$$

sapendo che gli autovalori della matrice $\tilde{\mathcal{B}}$ in (18) sono

$$\lambda_i^{(1,2)}(\tilde{\mathcal{B}}) = \frac{\sigma_{k+1} d_i \pm \sqrt{(\sigma_{k+1} d_i)^2 - 4\sigma_{k+1}\sigma_k}}{2},$$

dove $d_i = \frac{2}{e}(\lambda_i(A) - c)$ $i = 1, \dots, n$ e se $\tilde{b} > b \Longrightarrow$
$\tilde{x_k}$ della (17) converge all'autovettore principale di A.

L'Algoritmo 3.3 implementa la (17) con $A = LL^T$, [a,b]=[0,u_l] e $\tilde{b} = u_L$, dove $0 < u_l < \lambda_{max}(LL^T)$ e $u_l < u_L$.
Essendo che $\tilde{b}$ è impiegato solo per il riscalamento, si può usare $u_l = u_L$ per l'Algoritmo 3.3.
Osserviamo che qual'ora $u_l < u_L$ non sia preso per ipotesi, il fattore è $\rho_k = 1$, che corrisponde all'iterazione non scalata (11) e in questo caso si ha che il riscalo è fatto ai passo 22-23 dell'Algoritmo 3.4. Questa normalizzazione è fatta onde evitare overflow, soprattutto nel caso in cui m è piccolo. Possiamo dunque stabilire la convergenza dei metodi semplice e scalato tenendo conto di quanto detto sino ad ora:

Teorema 4.1. *Supponiamo che LL^T fine abbia un solo autovalore principale. Allora l'Algoritmo 3.4 produce un vettore che converge all'autovettore principale di LL^T.*

Dimostrazione. Definiamo gli estremi alla j-esima iterazione 0 e u_{l_j}, per la proprietà del quoziente di Rayleigh si ha che $\forall j \; 0 < u_{l_j} < \lambda_{max}(LL^T)$ e $u_{l_j} < u_{L_j}$.
Denotiamo inoltre $\mathcal{L}_j(t) = (t - \frac{u_{l_j}}{2})(2/u_{l_j})$. L'autovalore, unico, di modulo massimo è mappato nell'unico autovalore della matrice filtrata, che nel caso dell'algoritmo `Chebyshev_filter` è $Y = C_m(\mathcal{L}(LL^T))X$ mentre nel caso `Chebyshev_filter_scaled` è $Y = \frac{C_m(\frac{1}{e}(LL^T - eI))}{C_m(\frac{1}{e}(u_L - e))}X$. Allora applicando ripetutamente i Lemmi 4.2 e 4.3 si ha che l'Algoritmo converge all'autovettore principale di LL^T. ✓

Alla j-esima iterazione dell' Algoritmo 3.4 con estremi 0 e u_{l_j}, il gap-ratio della matrice filtrata dal polinomio di grado m, $C_m(\mathcal{L}_j(LL^T))$, è

$$\xi_{m_j} = \frac{\max_{i \neq 1} \| C_m(\frac{2}{u_{l_j}}(\lambda_i(LL^T) - \frac{u_{l_j}}{2}) \|}{C_m(\frac{2}{u_{l_j}}(\lambda_1(LL^T) - \frac{u_{l_j}}{2}))},$$

dove $\lambda_1(LL^T)$ è come assunto prima 'unico autovalore di modulo massimo. (Il filtaggio scalato ha lo stesso gap-ratio). Grazie alle proprietà di crescita fuori da [-1,1] del polinomio di Chebyshev di grado m, è semplice rendere ξ_{m_j} più piccolo del gap-ratio del metodo delle potenze. Se poniamo $\xi_m := \sup_j \xi_{m_j}$, si impiegano solo $O(\ln \tau / \ln \xi_m)$ passi dell'Algoritmo 3.4 per portare l'errore sotto la tolleranza data τ.

In teoria noi vorremmo che avere un u_{l_j} ottimale al passo j-esimo affinché il gap sia minimale, ma questo non è fattibile essendo gli autovalori sconosciuti. La combinazione convessa (6) è di fatto però effeciente, semplice e conveniente nella implementazione pratica.

5 Risultati Numerici

Per testare l'effettivo funzionamento del Metodo di Chebyshev e fare dei confronti col metodo delle potenze standard sono stati effettuati alcuni test su matrici di adiacenza dei grafi di alcune pagine Web. Nella seguente tabella sono riportati i dati di tali grafi.

Grafo	dimensione matrice d'adiacenza (n)	archi (nnz)	λ_2/λ_1
Stanford-Berkeley	685230	7600595	0.9999
wikipedia-20060925	2983494	37269096	0.7559
wikipedia-20061104	3148440	39383235	0.7733
wikipedia-20070206	3566907	45030389	0.8119

Tabella 1. Dati dei grafi.

Il gap-ratio per queste matrici, è stato calcolato tramite il comando `eigs`. Si nota che quello relativo alla matrice di Stanford-Berkeley è molto prossimo a 1, mentre per le altre è compreso tra 0.75 e 0.85.
I tests che seguono sono stati effettuati con Octave e Matlab senza riscontrare sostanziali differenza. Inotre è stato usato il filtro scalato, si è infatti notato che nella maggior parte dei casi i risultati coincidevano.

Riportiamo dunque quanto raccoloto delle sperimentazioni nelle seguenti tabelle dove sono specificati i vari parametri utilizzati: *f* è la norma residua, *m* il grado del polinomio di Chebyshev, *b* il parametro di convessità.

f	Cheb Cpu time (s)	MdP time (s)	Cheb iter	MdP iter	mpv Cheb	mpv MdP
1e-1	1.927	0.516	1	2	12	12
1e-2	10.87	0.748	15	3	180	18
1e-3	13.11	0.9488	18	4	216	24
1e-4	14.85	11.87	21	56	252	336
1e-5	15.94	24.57	23	116	276	696
1e-6	16.69	36.3	24	171	288	1026
1e-7	18.17	48.06	26	225	312	1350
1e-8	18.83	59.38	27	279	324	1674
1e-9	19.06	68.67	28	333	336	1998
1e-10	19.7	80.06	29	387	348	2322

Tabella A. Raccolta Cpu time dati e iterazioni con m=5, b=0.85, per il web-grafo Stanford-berkeley.

f	Cheb Cpu time (s)	MdP time (s)	Cheb iter	MdP iter	mpv Cheb	mpv MdP
1e-1	17.46	0.5132	21	2	252	12
1e-2	16.85	0.748	21	3	252	18
1e-3	16.74	0.9488	21	4	252	24
1e-4	16.94	11.87	21	56	252	336
1e-5	17.34	24.57	21	116	252	696
1e-6	16.79	36.3	21	171	252	1026
1e-7	16.84	48.06	21	225	352	1350
1e-8	552	59.38	745	279	8940	1674

Tabella B. Raccolta Cpu time dati e iterazioni con m=6, b=0.2 per il web-grafo Stanford-berkeley.

f	Cheb Cpu time (s)	MdP time (s)	Cheb iter	MdP iter	mpv Cheb	mpv MdP
1e-1	29.41	8.08	1	2	12	12
1e-2	39.55	20.85	2	6	24	36
1e-3	48.66	37.29	3	11	36	66
1e-4	57.61	50.44	4	15	48	90
1e-5	58.42	65.72	4	19	48	114
1e-6	69.75	80.49	5	23	60	138
1e-7	70.05	92.52	5	27	60	162
1e-8	78.75	107.3	6	31	72	186
1e-9	79.62	118	6	35	72	210
1e-10	81.33	135.4	6	39	72	234

Tabella C. Raccolta Cpu time dati e iterazioni con m=5, b=0.85, per il web-grafo wikipedia-20060925.

f	Cheb Cpu time (s)	MdP time (s)	Cheb iter	MdP iter	mpv Cheb	mpv MdP
1e-1	31.61	8.969	1	2	12	12
1e-2	40.55	22.28	2	6	24	36
1e-3	50.66	39.85	3	11	36	66
1e-4	63.13	53.38	4	15	48	90
1e-5	72.17	77.74	5	20	60	120
1e-6	77.65	92.08	5	24	60	144
1e-7	90.08	115	6	29	72	174
1e-8	89.48	129.9	6	33	72	198
1e-9	102.1	145.2	7	38	84	228
1e-10	103	162	7	42	84	252

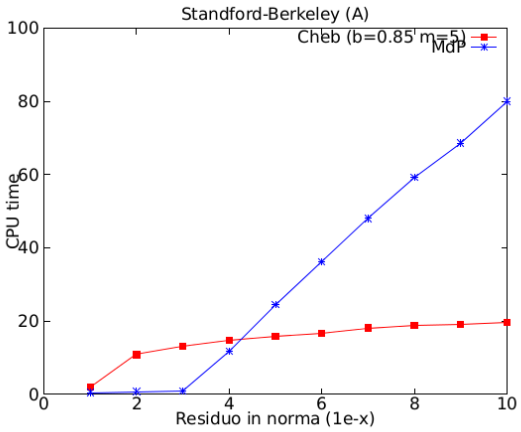
Tabella D. Raccolta Cpu time dati e iterazioni con m=5, b=0.85, per il web-grafo wikipedia-20061104.

f	Cheb Cpu time (s)	MdP time (s)	Cheb iter	MdP iter	mpv Cheb	mpv MdP
1e-1	81.4	22.7	1	2	12	12
1e-2	111.7	51.12	2	5	24	30
1e-3	139.4	107.1	3	11	36	66
1e-4	199.5	150.8	4	16	48	96
1e-5	192.7	211	5	22	60	132
1e-6	230.9	250.9	6	27	72	162
1e-7	246.8	308.3	7	33	84	198
1e-8	254.5	361.3	7	39	84	234
1e-9	276.6	396	8	44	96	264
1e-10	265.3	440.4	8	50	96	300

Tabella E. Raccolta Cpu time dati e iterazioni con m=5, b=0.85, per il web-grafo wikipedia-20070206

Si nota che al diminuire del valore residuo in norma il metodo di Chebyshev è sensibimente più veloce, si arriva persino a dimezzare i tempi. Il numero di moltiplicazioni matrice vettore è calcolato utilizzando il fatto che per ogni iterzione si hanno 12 prodotti per il metodo di Chebyshev e 6 per il metodo delle potenze.
Per quanto riguarda la procisione del metodo di ha che generalmente la distanza, in media, dell'entrate degli autovettori generati C e P, rispettivamente per il metodo di Chebyshev e quello delle potenze, $\vartheta = \frac{\|C-P\|_1}{n}$ è dell'ordine di 1^{-10}.

Immagine 1. *Comparazione grafica dei dati raccolti del grafo Stanford-Berkeley.*



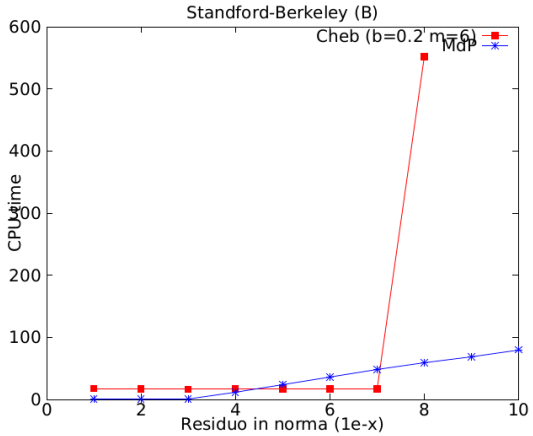
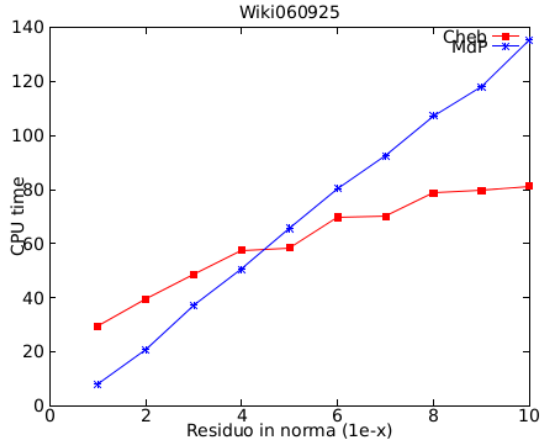
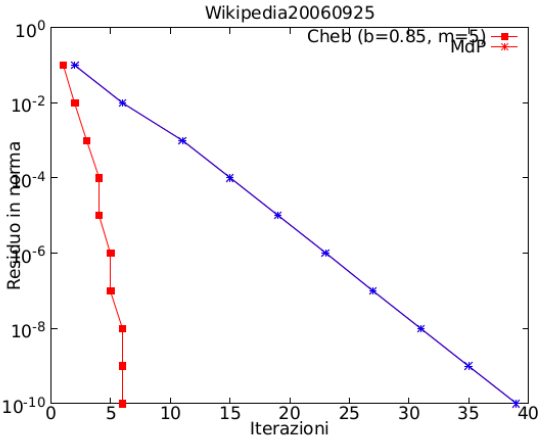


Immagine 2. *Comparazione grafica dei dati raccolti del grafo wikipedia-20060925.*





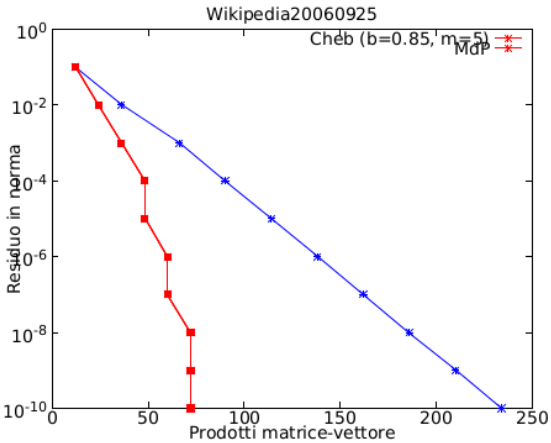
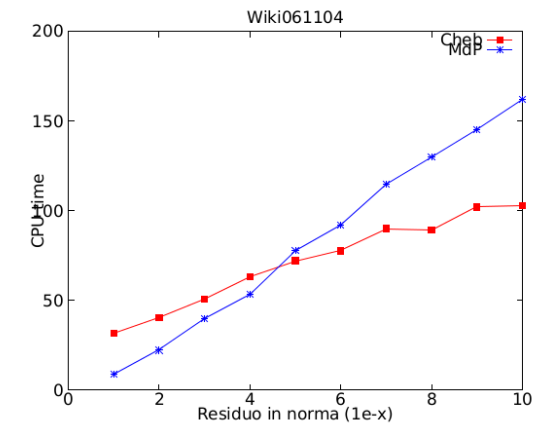
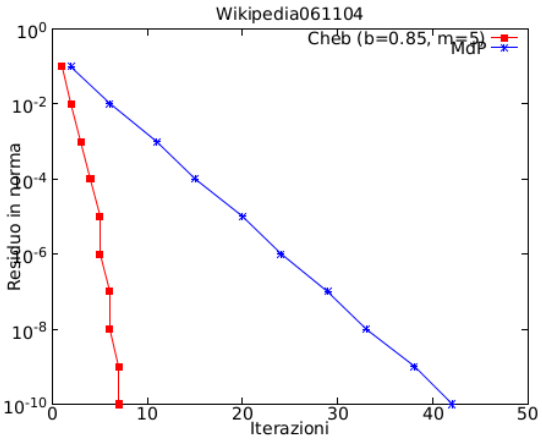


Immagine 3. *Comparazione grafica dei dati raccolti del grafo wikipedia-20061104.*





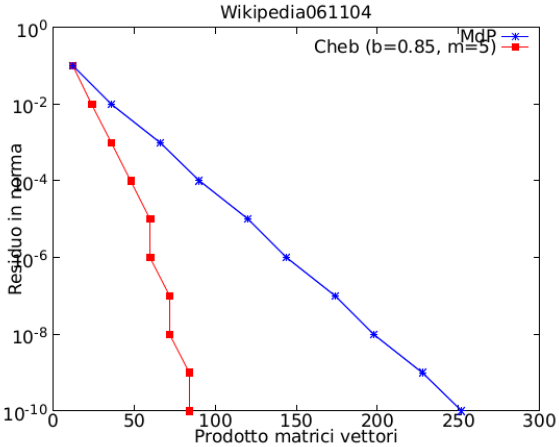
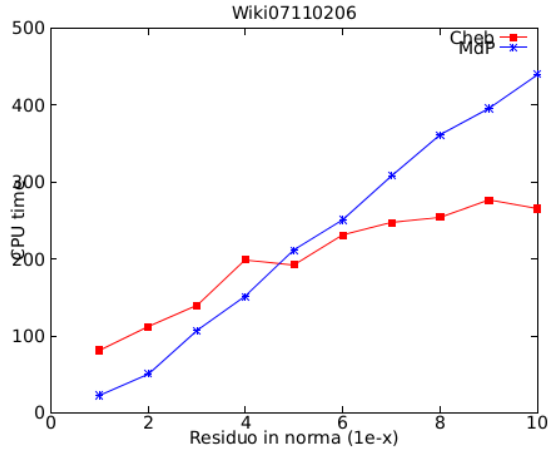
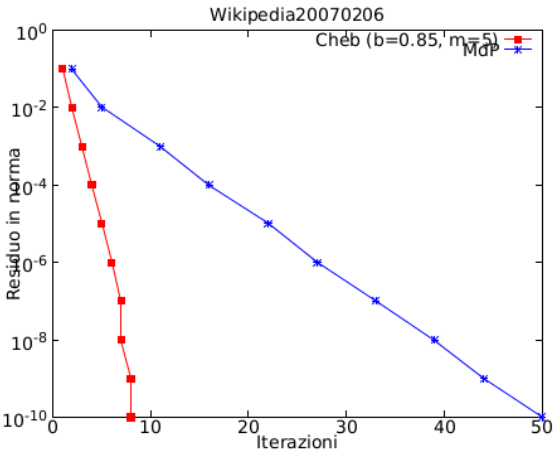
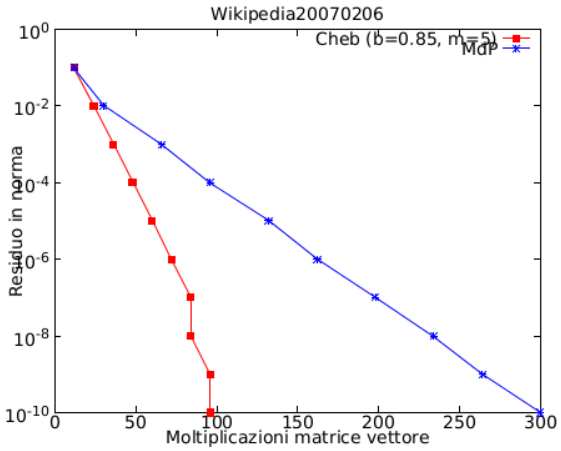


Immagine 4. *Comparazione grafica dei dati raccolti del grafo wikipedia-20070206.*







5.1 Nota

Le sperimentazioni fatte sui dati risultano generalmente coerenti con quanto riportato nell'articolo, in tutti i casi infatti si verifica che $m = 5$ e $b = 0.85$ sono parametri ottimali. Fa eccezione il grafo di Stanford-Berkeley, per cui sono indicati nell'articolo come valori ottimali $m = 6$ e $b = 0.2$, ma dalle prove effetuate a partire dal valore di valutazione del passo 10^{-8} il numero di iterazioni balza all'ordine delle migliaia, mentre risulta avere convergenza buona con i parametri $m = 5$ e $b = 0.85$.

6 Note teoriche

Ecco una piccola appendice contenente i risultati teorici citati nelle sezioni precedenti a cui si ritiene opportuno un richiamo.

Definizione 6.1. *Si dice, data $A \in \mathcal{M}(N)$ e $b \in \mathbb{R}^N$* **sottospazio di Krylov**

$$\mathcal{K}_R = Span(b,\ Ab,\ A^2b, \dots\ A^{R-1}b)$$

Definizione 6.2. *Data una matrice hermitiana A non nulla il* **quoziente di Rayleigh-Ritz** *per x vettore nonnullo è $\frac{x^HAx}{x^Hx} = \frac{\langle Ax,x\rangle}{\langle x,x\rangle}$.*

Definizione 6.3. *È detto* **metodo di approssimazione di Lanczos** *il metodo per la ricerca di autocoppie di una matrice A che sfrutta la* **decomposizione di Lanczos**

$$AV_k = V_kT_k + f_ke_k^T,\ V_k^TV_k = I_k \text{ e } V_k^Tf_k = \mathbf{0}$$

dove V_k è la relativa ad una base ortonormale del sottospazio di Krylov $\mathcal{K}_k(A, v)$.
Si definisceinoltre la **coppia valore-vettore di Ritz**, *date $(\lambda_i, x_i)\,\forall\ i = 1, \dots, k$ le autocoppie di T_k i λ_i sono detti* **valori di Ritz** *e i $v_i = V_kx_i$ sono detti* **vettori di Ritz**.
Vale che $y^t(Av - \lambda v) = 0$.

Teorema 6.1 (**Teorema di Ritz-Rayleigh**). *Sia $A \in \mathcal{M}(n)$ matrice Hermitiana e siano $\lambda_{min} = \lambda_1 < \lambda_2 < \dots < \lambda_n = \lambda_{max}$ i relativi autovalori $\Longrightarrow$*

$$\lambda_1x^Hx \le x^HAx \le \lambda_nx^Hx\ \forall\ x \in \mathbb{C}^n$$

$$\lambda_{max} = \max_{x\neq 0}\frac{x^HAx}{x^Hx} = \max_{x\neq 0}\ R(A,x) = \max_{x^Hx=1}\ x^HAx$$

$$\lambda_{min} = \min_{x \neq 0} \frac{x^H A x}{x^H x} = \min_{x \neq 0} R(A, x) = \min_{x^H x = 1} x^H A x$$

Teorema 6.2 (**Teorema minmax di Courant-Fisher**). *Sia* $A \in \mathcal{S}(n) \Rightarrow \lambda_k(A) = \max_{dim(S)=k} \min_{y \in S \setminus 0} \frac{x^t A x}{x^t x}$.

Teorema 6.3 (**Teorema di Perron-Frobenius**). *Sia data* $A \in \mathcal{M}(n)$ *matrice tale che* $\forall\ (i,j)\, a_{i,j} \geq 0 \Rightarrow \exists x, y$ *autovettori destro e sinistro di* $\lambda = \rho(A)$.
Inotre se A *è irriducibile* λ *è semplice e gli autovettori hanno componenti positive.*
Infine se le entrate di A *sono strettamente positive* λ *è l'unico autovalore di modulo massimo.*

Riportiamo l'algoritmo per il **metodo delle potenze** implementato:

Algoritmo 6.1. .

```
function [lambda,x,iter]=MetPotenze(A,tol,kmax,x0)
tic
x0=x0/norm(x0);
lambda=x0'*(A*((A')*x0));
err=tol*abs(lambda)+1;
iter=0;
while (err > tol*abs(lambda) & abs(lambda~=0 & iter<=kmax)
x=A*((A')*x0);
x=x/norm(x);
lamnew=x'*(A*((A')*x));
err=abs(lamnew-lambda);
lambda=lamnew;
x0=x;
iter=iter+1;
end
toc
end
```