Analisi dell'algoritmo Arnoldi-PET per il calcolo del PageRank

Letizia D'Achille

sulla base dell'articolo
"A variant of the Power–Arnoldi algorithm for computing PageRank"

1 giugno 2021

Introduzione

Nella seguente relazione si prende in considerazione il metodo Arnoldi-PET studiato da Qian-Ying Hu et al. [1] per la risoluzione del problema del PageRank. Si riprodurranno quindi le sperimentazioni effettuate nell'articolo al fine di confermarne i risultati, e se ne effettueranno di altre per studiare il comportamento dell'algoritmo al variare dei parametri e della matrice utilizzata. È utile a tal fine introdurre brevemente il problema.

Il PageRank è un metodo utilizzato per l'ordinamento delle pagine web, che ne determina l'*importanza* basandosi sulla rete di collegamenti ipertestuali che le collega.
Tale struttura di grafo si modella con una matrice di adiacenza H, dalla quale si rimuovono i *dangling nodes* considerando le righe identicamente nulle e sostituendo ad esse delle righe di elementi uguali a 1. Quindi si costruisce la matrice diagonale $D = (d_{ij})$ con elementi d_{ii} che corrispondono alla somma delle righe di H.
In questo modo si costruisce la matrice

$$A = \alpha H^T D^{-1} + (1 - \alpha) v e^T$$

dove $\alpha \in (0, 1)$ è detto *damping factor*, e è un vettore di elementi uguali a 1 e v è un vettore di elementi positivi tale che $v^T e = 1$, detto vettore di personalizzazione. La matrice A soddisfa le ipotesi del teorema di Perron-Frobenius, qui richiamato.

Teorema (Perron-Frobenius)**.** *Sia $A = (a_{ij})_{ij} \in \mathcal{M}(\mathbb{R}, n)$, allora valgono le seguenti affermazioni*

- *se $A \geq 0$ allora $\exists \lambda \in Sp(A)$ tale che $\lambda = \rho(A)$ ed in corrispondenza un autovettore destro x e uno sinistro y^T tali che $x_i \geq 0$ e $y_i \geq 0$ $\forall i = 1, \ldots, n$;*

- *se $A \geq 0$ ed irriducibile allora $\rho(A)$ è un autovalore semplice e gli autovettori destro e sinistro sono strettamente positivi;*

- *se $A > 0$ allora $\rho(A)$ è l'unico autovalore di modulo massimo.*

Denotando $P = H^T D^{-1}$, si ha che tale matrice è stocastica rispetto alle colonne (la somma di ciascuna colonna è 1) o, equivalentemente, e^T ne è autovettore sinistro rispetto all'autovalore 1. Segue che vale la stessa proprietà per la matrice A, e da questo si ottiene $\rho(A) = 1$.
Per cui dal teorema di Perron-Frobenius, poichè A è a elementi positivi, segue che ha come autovalore 1, unico di modulo massimo e semplice. Si definisce quindi il corrispondente autovettore x, unico a meno di normalizzazione, tale che $Ax = x$. Questo è detto vettore del PageRank ed ha come elementi le *importanze* che vengono assegnate alle pagine web.

I metodi discussi nella relazione hanno come fine l'approssimazione del vettore del PageRank, e trattano il problema nella forma di ricerca di un autovettore. Si vuole prestare particolare attenzione al metodo Arnoldi-PET, e al confronto con i due metodi da cui discende, ovvero il metodo PET e il metodo thick restarted Arnoldi. Inoltre si condurrà un confronto con il metodo Power-Arnoldi, basato sul metodo delle potenze, sviluppato precedentemente [2].

Metodi

PET

Il metodo delle potenze con estrapolazione incentrata sulla traccia della matrice (PET) [8] si basa sul metodo delle potenze; sfrutta la forma dei polinomi minimo e caratteristico della matrice per ottenere un algoritmo più efficiente.

Come si è già osservato, se si denota con $Sp(A) = \{\lambda_1, \ldots, \lambda_n\}$ lo spettro di A, per il teorema di Perron-Frobenius la matrice soddisfa $1 = |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| \geq 0$. Di conseguenza è possibile scrivere il polinomio caratteristico di A come

$$p(\lambda) = (\lambda - 1)p_1(\lambda) \quad \text{con} \quad (\lambda - 1) \nmid p_1(\lambda).$$

Se si considera l'ideale $I = \{f(\lambda) \in \mathbb{K}[\lambda] | f(A) = 0\}$ generato dal polinomio minimo $q(\lambda)$, allora per il teorema di Cayley-Hamilton vale $p(\lambda) \in I$. Segue quindi $(A - I)p_1(A) = 0$, ovvero $(A - I)p_1(A)u = 0 \ \forall u \in \mathbb{R}^n$.
Allo stesso tempo deve valere $p_1(A) \neq 0$ in quanto altrimenti $p_1(\lambda) \in I$ e varrebbe $q(\lambda)|p_1(\lambda)$, ma $q(\lambda)$ contiene il fattore $(\lambda - 1)$ che non divide $p_1(\lambda)$. Esiste quindi un vettore non nullo u_0 tale che c. Per il vettore x così definito vale $(A - I)x = 0$, ovvero x è il vettore del PageRank.

Non è tuttavia possibile calcolare direttamente $p_1(A)u_0$ essendo computazionalmente gravoso. È possibile tuttavia approssimare $p_1(\lambda)$ sapendo che

$$p_1(\lambda) = \lambda^{n-1} - (\mu - 1)\lambda^{n-2} + \alpha_3\lambda^{n-3} + \cdots + \alpha_p\lambda^{n-p}$$

con $p \leq n - l + 1$ dove μ è la traccia della matrice A, l il numero dei dangling nodes e $\alpha_i \quad i = 3, \ldots, p$ sono scalari con $\alpha_p \neq 0$.
Tale scrittura del polinomio $p_1(\lambda)$ è possibile in quanto 0 ha molteplicità almeno $l - 1$ nel polinomio caratteristico di A (da cui la disuguaglianza per p), infatti $Ker(A)$ ha dimensione almeno $l - 1$. Questo segue dal fatto che la matrice A ha l righe uguali poiché la stessa P ne ha l uguali in corrispondenza dei dangling nodes (tutti gli elementi sono uguali a $1/n$).

Si può usare $\lambda^{n-1} - (\mu - 1)\lambda^{n-2}$ come approssimazione per $p_1(\lambda)$, da cui il vettore $[A^{n-1} - (\mu - 1)A^{n-2}]u_0$ può essere considerato una approssimazione del vettore del PageRank. Si ha inoltre

$$\begin{aligned}
[A^{n-1} - (\mu - 1)A^{n-2}]u_0 &= A^{n-m_1-1}[A - (\mu - 1)]A^{m_1-1}u_0 \\
&= A^{n-m_1-1}[A - (\mu - 1)]u_{m_1-1} \\
&= A^{n-m_1-1}[u_{m_1} - (\mu - 1)u_{m_1-1}]
\end{aligned}$$

dove u_i è il vettore che si ottiene all'iterazione i del metodo delle potenze, e $m_1 < n - 1$ è un intero positivo.

Queste osservazioni suggeriscono quindi un metodo di estrapolazione basato sulla traccia e sui due vettori u_{m_1-1}, u_{m_1} che si ottengono al passo m_1 del metodo delle potenze. Il PET si svolge quindi usando quest'ultimo metodo per m_1 passi, poi si esegue l'estrapolazione ricavando il nuovo vettore $u_0 = u_{m_1} - (\mu - 1)u_{m_1-1}$ che viene utilizzato come vettore di partenza per ulteriori m_1 step del metodo delle potenze.
Iterando il procedimento si ottiene infine un'approssimazione del vettore del PageRank. L'algoritmo è presentato in Figura 9.

Note sull'implementazione La formula per il calcolo della traccia μ della matrice A si può verificare facilmente dopo aver osservato che, ad eccezione delle posizioni corrispondenti ai dangling nodes, sulla diagonale di P ci siano elementi nulli. Questo è vero in quanto sulla diagonale di H ci sono 0 (come convenzione una pagina non ha link verso se stessa) e $P = H^T D^{-1}$. Si dimostra allora

$$\begin{aligned}
\mu = tr(A) &= \alpha\ tr(P) + (1 - \alpha)\ tr(ve^T) \\
&= \alpha tr(P) + (1 - \alpha) \\
&= 1 + \alpha(tr(P) - 1) \\
&= 1 + \alpha\left(\frac{l}{n} - 1\right).
\end{aligned}$$

Thick restarted Arnoldi

Si descrive ora brevemente l'iterazione di Arnoldi utile all'algoritmo thick restarted Arnoldi. Tale processo costruisce una base ortonormale $\{v_1, v_2, \ldots, v_n\}$ del sottospazio di Krylov di dimensione m

$$\mathcal{K}_m(A, v_1) = span(v_1, Av_1, \ldots, A^{m-1}v_1).$$

L'iterazione costruisce $V_m = [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n \times m}$, matrice che ha come colonne la base ortonormale di $\mathcal{K}_m(A, v_1)$, e una matrice di Hessenberg superiore

$$\overline{H}_m = \begin{pmatrix} H_m \\ h_{m+1,m} e_m^T \end{pmatrix} \in \mathbb{R}^{(m+1) \times m}$$

con $H_m \in \mathbb{R}^{m \times m}$ a sua volta di Hessenberg superiore.
È possibile verificare che vale la seguente relazione

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T = V_{m+1} \overline{H}_m.$$

L'algoritmo è presentato in Figura 15.

Per migliorare i costi in spazio dell'iterazione di Arnoldi all'aumentare del valore di m, corrispondente alla dimensione del sottospazio di Krylov voluto, si utilizza la tecnica del *restart* in modo che m rimanga contenuto.
In particolare l'algoritmo thick restarted Arnoldi [7, 9] implementa una tecnica di restart che ottimizza l'iterazione di Arnoldi costruendo in anticipo una parte delle matrici V_m e $\overline{H}_m$.

A questo scopo si sceglie un intero $p < m$ che denota il numero di coppie autovalore-autovettore da approssimare. Vengono quindi costruite la matrice V_{p+1}, che andrà a costituire le prime $p + 1$ colonne di V_m, e la matrice $\overline{H}_p$, che andrà a formare la sottomatrice di testa di $\overline{H}_m$ di dimensione $(p + 1) \times p$.
Tali matrici sono costruite in modo che sia conservata la relazione

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T = V_{m+1} \overline{H}_m.$$

L'algoritmo è presentato in Figura 11.

Note sull'implementazione Si può osservare in particolare che $\overline{H}_p$ non deve necessariamente essere di Hessenberg superiore, di conseguenza non lo sarà neanche $\overline{H}_m$. Inoltre il processo di ortogonalizzazione è stato eseguito con l'algoritmo di Gram-Schmidt (il cui codice è presentato in Figura 16). L'implementazione con la funzione *orth* fornita da MATLAB non è adatta in quanto genera una base ortonormale tramite la decomposizione in valori singolari, che non assicura la conservazione della relazione tra le matrici V_{m+1} e $\overline{H}_m$.

Arnoldi-PET

Il metodo Arnoldi-PET combina i due metodi presentati precedentemente al fine di migliorarne le prestazioni. Il metodo PET presenta infatti una convergenza lenta per valori alti del damping factor α, mentre la singola iterazione risulta molto veloce. D'altro canto l'algoritmo thick restarted Arnoldi converge in meno iterazioni, ma l'esecuzione di un passo è computazionalmente più pesante.

La fusione dei due metodi avviene quindi nel seguente modo: inizialmente si eseguono poche iterazioni dell'algoritmo thick restarted Arnoldi per ottenere un'approssimazione del vettore del PageRank a partire da un vettore iniziale x_0; se l'approssimazione non è soddisfacente, essa viene utilizzata come vettore iniziale per una breve iterazione del metodo PET; se il nuovo vettore ancora non raggiunge l'accuratezza richiesta, lo si usa come vettore iniziale per ripetere l'algoritmo thick restarted Arnoldi.
Il procedimento viene iterato fino a che l'errore non scende sotto la tolleranza richiesta.

Il passaggio dal metodo PET all'algoritmo basato sull'iterazione di Arnoldi sfrutta i tre parametri β, *restart* e *maxit*.
Si usa il parametro *maxit* per controllare il numero di iterazioni del metodo PET. Tali iterazioni sono conteggiate parzialmente dal parametro *restart*, che viene incrementato di 1 quando il rapporto tra la norma del residuo del passo corrente e quella del residuo del passo precedente diviene maggiore del terzo parametro β.
Si effettua quindi il passaggio dal metodo PET all'algoritmo thick restarted Arnoldi nel momento in cui *restart* supera *maxit*.

È noto che la convergenza asintotica del metodo delle potenze dipende dal rapporto $\frac{|\lambda_2|}{|\lambda_1|}$ [10], che nel caso del problema del PageRank corrisponde al solo $|\lambda_2|$. Sapendo in particolare che $|\lambda_2| \leq \alpha$ per il Teorema 2 (che verrà dimostrato nella sezione che segue), si ha che il tasso di convergenza del metodo delle potenze è limitato superiormente dal damping factor α.
Date queste osservazioni ha senso porre il parametro β più piccolo di α, in particolare sarà usato $\beta = \alpha - 0.1$.

L'algoritmo è presentato in Figura 13.

Note sull'implementazione Le procedure *short_tra_fr* e *short_tra* sono versioni dell'algoritmo thick restarted Arnoldi in cui ne vengono eseguite poche iterazioni, nello specifico due iterazioni, come previsto dal metodo Arnoldi-PET. In particolare *short_tra_fr* viene eseguita solo alla prima iterazione del metodo Arnoldi-PET, mentre *short_tra* in tutte quelle che seguono. Come descritto nei commenti all'algoritmo, la prima esegue i passi 2-7 dell'algoritmo thick restarted Arnoldi, mentre la seconda esegue solo i passi 3-7. Differiscono quindi unicamente nel fatto che *short_tra_fr* non esegue il controllo sulla norma del residuo alla prima iterazione, a differenza di quanto accade in *short_tra*.

Convergenza del metodo Arnoldi-PET

Si enunciano inizialmente due teoremi utili a descrivere lo spettro della matrice A. Il seguente è una specializzazione del teorema di Brauer nel caso particolare del problema del PageRank. Nella notazione si suppone sempre $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

Teorema 1 (Brauer [4])**.** *Si assuma che lo spettro di una matrice P stocastica rispetto alle colonne sia $\{1, \gamma_2, \ldots, \gamma_n\}$, allora lo spettro della matrice $A = \alpha P + (1-\alpha)ve^T$ è $\{\lambda_1, \lambda_2, \ldots, \lambda_n\} = \{1, \alpha\gamma_2, \ldots, \alpha\gamma_n\}$, dove $0 < \alpha < 1$ e v è un vettore con elementi non-negativi tale che $e^T v = 1$.*

Dimostrazione. Dato che P è stocastica si ha che e è un autovettore di P relativo all'autovalore 1. Si consideri una matrice invertibile $Q = \begin{pmatrix} e & X \end{pmatrix}$, e la sua inversa $Q^{-1} = \begin{pmatrix} y^T \\ Y^T \end{pmatrix}$. Allora

$$\begin{pmatrix} 1 & 0 \\ 0 & I \end{pmatrix} = Q^{-1}Q = \begin{pmatrix} y^T e & y^T X \\ Y^T e & Y^T X \end{pmatrix}$$

da cui le due identità $y^T e = 1$ e $Y^T e = 0$.
Considerando

$$Q^{-1}PQ = \begin{pmatrix} y^T e & y^T PX \\ Y^T e & Y^T PX \end{pmatrix} = \begin{pmatrix} 1 & y^T PX \\ 0 & Y^T PX \end{pmatrix}$$

ho che per similitudine $Y^T PX$ ha come autovalori i rimanenti $\gamma_2, \ldots, \gamma_n$ di P. Applicando la stessa trasformazione ad $A = \alpha P + (1-\alpha)ve^T$ si ottiene

$$\begin{aligned} Q^{-1}(\alpha P + (1-\alpha)ve^T)Q &= \alpha Q^{-1}PQ + (1-\alpha)Q^{-1}ev^T Q \\ &= \begin{pmatrix} \alpha & \alpha y^T PX \\ 0 & \alpha Y^T PX \end{pmatrix} + (1-\alpha)\begin{pmatrix} y^T e \\ Y^T e \end{pmatrix}\begin{pmatrix} v^T e & v^T X \end{pmatrix} \\ &= \begin{pmatrix} \alpha & \alpha y^T PX \\ 0 & \alpha Y^T PX \end{pmatrix} + \begin{pmatrix} (1-\alpha) & (1-\alpha)v^T X \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & \alpha y^T PX + (1-\alpha)v^T X \\ 0 & \alpha Y^T PX \end{pmatrix} \end{aligned}$$

da cui segue per similitudine che gli autovalori di A sono $\{1, \alpha\gamma_2, \ldots, \alpha\gamma_n\}$. □

Teorema 2 ([5])**.** *Sia P una matrice stocastica rispetto alle colonne. Sia α un numero reale tale che $0 < \alpha < 1$ e v un vettore con elementi non-negativi tale che $e^T v = 1$. Se $A = \alpha P + (1-\alpha)ve^T$, allora vale $\lambda_1 = 1$, $|\lambda_2| \leq \alpha$.*

Dimostrazione. Dato che P è non-negativa si può applicare il primo punto del Teorema di Perron-Frobenius, per cui $\exists \gamma \in Sp(P)$ tale che $\gamma = \rho(P)$. Dato che è stocastica si ha che $\rho(P) = 1$, per cui $Sp(P) = \{1, \gamma_2, \ldots, \gamma_n\}$ con $|\gamma_2| \leq 1$. Per il Teorema 1 segue allora che $Sp(A) = \{1, \alpha\gamma_2, \ldots, \alpha\gamma_n\}$ con $|\lambda_2| = |\alpha\gamma_2| \leq |\alpha|$. □

L'analisi della convergenza del metodo Arnoldi-PET si concentra sul passaggio dal metodo PET all'algoritmo thick restarted Arnoldi. Si effettua quindi un confronto tra il metodo Arnoldi e il metodo Arnoldi-PET. Di seguito si enunciano due risultati preliminari.

Lemma 1 ([6]). *Il sottospazio di Krylov* $\mathcal{K}_m \equiv \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\}$ *è il sottospazio di tutti i vettori in* $\mathbb{C}^n$ *che possono essere scritti come* $x = p(A)v$*, dove* p *è un polinomio di grado al più* $m-1$*.*

Lemma 2 ([6]). *Sia* P *un proiettore ortogonale sul sottospazio* S*. Allora per ogni vettore* $x \in \mathbb{C}^n$ *vale*

$$\|x - Px\|_2 = \min_{y \in S} \|x - y\|_2$$

Sia $\mathcal{L}_{m-1}$ l'insieme dei polinomi di grado al più $m-1$. Si mostra ora un risultato di convergenza del metodo Arnoldi, che risulterà migliorato dall'Arnoldi-PET.

Teorema 3 ([6]). *Si assuma che* A *sia diagonalizzabile e che il vettore iniziale* v_1 *nel metodo di Arnoldi ammetta la scrittura* $v_1 = \sum_{i=1}^n \gamma_i x_i$ *rispetto alla base di autovettori* $\{x_i\}_{i=1,\ldots,n}$*, nella quale* $\|x_i\|_2 = 1$*,* $i = 1, \ldots, n$ *e* $\gamma_1 \neq 0$*. Allora vale la seguente disuguaglianza*

$$\|(I - \mathscr{P}_m)x_1\|_2 \leq \xi \varepsilon_m$$

dove $\mathscr{P}_m$ *è il proiettore ortogonale sul sottospazio* $\mathcal{K}_m(A, v_1)$*,* $\xi = \sum_{j=2}^n \frac{|\gamma_j|}{|\gamma_1|}$ *e*

$$\varepsilon_m = \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \max_{\lambda \in Sp(A) \setminus \lambda_1} |p(\lambda)|.$$

Dimostrazione. Si ha che

$$\begin{aligned}
\|(I - \mathscr{P}_m)\gamma_1 x_1\|_2 &= \min_{x \in \mathcal{K}_m} \|\gamma_1 x_1 - x\|_2 \\
&= \min_{p \in \mathcal{L}_{m-1}} \|\gamma_1 x_1 - p(A)v_1\|_2 \\
&\leq \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \|\gamma_1 x_1 - p(A)v_1\|_2.
\end{aligned}$$

dove la prima uguaglianza segue dal Lemma 2 e la seconda dal Lemma 1. Scrivendo quindi v nella base $\{x_j\}_{j=1,\ldots,n}$ segue

$$\begin{aligned}
\|(I - \mathscr{P}_m)\gamma_1 x_1\|_2 &\leq \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \|\gamma_1 x_1 - p(A) \sum_{j=1}^n \gamma_j x_j\|_2 \\
&= \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \|\gamma_1 x_1 - \sum_{j=1}^n \gamma_j p(A) x_j\|_2 \\
&= \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \|\sum_{j=2}^n \gamma_j p(\lambda_j) x_j\|_2 \\
&\leq \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \max_{\lambda \in Sp(A) \setminus \lambda_1} |p(\lambda)| \sum_{j=2}^n |\gamma_j|
\end{aligned}$$

dove la seconda uguaglianza segue dal fatto che $\{x_j\}_{j=1,\dots,n}$ è una base di autovettori e i polinomi considerati sono tali che $p(\lambda_1) = 1$; nell'ultimo passaggio viene usata la disuguaglianza triangolare e il fatto che $\|x_j\|_2 = 1,\ j = 1, \dots, n$. Il risultato segue dividendo entrambi i membri per $|\gamma_1|$. □

È quindi utile fornire la formula che deriva dall'iterazione del metodo PET.

Lemma 3. *Sia v_1 il vettore iniziale per il metodo PET, preso dalla precedente esecuzione del metodo thick restarted Arnoldi. Allora l'iterazione PET nel metodo Arnoldi-PET produce il vettore*

$$v_1^{new} = \eta G^k v_1 \qquad G = A^{m_1-1}[A - (\mu - 1)I] \tag{1}$$

dove $k \geq$ maxit, η è il fattore normalizzante, μ è la traccia della matrice A, m_1 è il numero di iterazioni del metodo delle potenze, G è detta matrice di iterazione e I è l'identità $n \times n$.

Dimostrazione. Dopo l'esecuzione di m_1 iterazioni del metodo delle potenze, si ottengono

$$x^{(m_1)} = A^{m_1} v_1 \ ,\ x^{(m_1-1)} = A^{m_1-1} v_1.$$

Usando lo schema di estrapolazione basato su $x^{(m_1-1)}$, $x^{(m_1)}$ e μ, si ottiene

$$\begin{aligned} \hat{v_1} &= x^{(m_1)} - (\mu - 1)x^{(m_1-1)} = A^{m_1} v_1 - (\mu - 1)A^{m_1-1} v_1 \\ &= A^{m_1-1}[A - (\mu - 1)]v_1 = Gv_1. \end{aligned}$$

Per il comportamento del metodo Arnoldi-PET si ha che l'iterazione del PET viene eseguita k volte con $k \geq$ maxit, da cui segue la tesi. □

Dati i teoremi precedenti, è ora possibile mostrare la disuguaglianza che caratterizza il metodo Arnoldi-PET. Si suppone quindi che il vettore v_1^{new} sia il vettore iniziale per l'algoritmo thick restarted Arnoldi. Secondo l'algoritmo sarà quindi il vettore di partenza per un'iterazione di Arnoldi di m passi, che costruisce il sottospazio di Krylov $\mathcal{K}_m(A, v_1^{\text{new}}) = \text{span}\{v_1^{\text{new}}, Av_1^{\text{new}}, \dots, A^{m-1}v_1^{\text{new}}\}$. Il teorema che segue mostra quindi il miglioramento della convergenza del metodo Arnoldi ottenuto grazie all'uso di un vettore di partenza proveniente dal metodo PET.

Teorema 4. *Si assuma che la matrice A sia diagonalizzabile, e si denoti con $\tilde{\mathscr{P}}_m$ il proiettore ortogonale sul sottospazio $\mathcal{K}_m(A, v_1^{new})$. Allora con le notazioni precedenti vale la seguente disuguaglianza*

$$\|(I - \tilde{\mathscr{P}}_m)x_1\|_2 \leq \alpha^{k(m_1-1)} \left(\frac{\alpha + 1 - \mu}{2 - \mu}\right)^k \cdot \xi\varepsilon_m$$

dove $k \geq$ maxit.

Dimostrazione. Usando il risultato (1) e il Lemma 1, si ha che per ogni $u \in \mathcal{K}_m(A, v_1^{\text{new}})$, esiste $q(x) \in \mathcal{L}_{m-1}$ tale che

$$\begin{aligned} u = q(A)v_1^{\text{new}} &= \eta q(A)G^k v_1 = \eta q(A)G^k \left(\gamma_1 x_1 + \sum_{j=2}^{n} \gamma_j x_j \right) \\ &= \eta\gamma_1 q(A)G^k x_1 + \eta q(A) \sum_{j=2}^{n} \gamma_j G^k x_j \end{aligned} \tag{2}$$

dove $v_1 = \sum_{j=1}^{n} \gamma_j x_j$ è la scrittura di v_1 rispetto alla base di autovettori $\{x_i\}_{i=1,\ldots,n}$. Usando la matrice di iterazione in (1) e i fatti $Ax_1 = x_1$, $Ax_j = \lambda_j x_j$ $(j = 2, \ldots, n)$, si ha

$$\begin{aligned} Gx_j = A^{m_1-1}[A - (\mu-1)I]x_j &= A^{m_1}x_j - (\mu-1)A^{m_1-1}x_j \\ &= \begin{cases} (2-\mu)x_1 & j = 1 \\ \lambda^{m_1-1}(\lambda_j - \mu + 1)x_j & j = 2, \ldots, n \end{cases} \end{aligned} \tag{3}$$

Sostituendo il risultato di (3) in (2), si ottiene

$$u = \eta\gamma_1 q(1)(2-\mu)^k x_1 + \eta \sum_{j=2}^{n} \gamma_j q(\lambda_j)[\lambda_j^{m_1-1}(\lambda_j - \mu + 1)]^k x_j$$

e quindi

$$\begin{aligned} \left\| \frac{u}{\eta\gamma_1 q(1)(2-\mu)^k} - x_1 \right\|_2 &= \left\| \sum_{j=2}^{n} \frac{\gamma_j}{\gamma_1} \frac{q(\lambda_j)}{q(1)} \left[\frac{\lambda_j^{m_1-1}(\lambda_j - \mu + 1)}{2-\mu} \right]^k x_j \right\|_2 \\ &\leq \left[\frac{\alpha^{m_1-1}(\alpha + 1 - \mu)}{2-\mu} \right]^k \cdot \sum_{j=2}^{n} \frac{|\gamma_j|}{|\gamma_1|} \frac{|q(\lambda_j)|}{|q(1)|} \\ &\leq \alpha^{k(m_1-1)} \left(\frac{\alpha + 1 - \mu}{2-\mu} \right)^k \cdot \sum_{j=2}^{n} \frac{|\gamma_j|}{|\gamma_1|} \max_{j \neq 1} |p(\lambda_j)| \end{aligned}$$

dove si è usato $|\lambda_2| \leq \alpha$, la relazione $|\lambda_j - \mu + 1| \leq |\lambda_j| + |1 - \mu| = |\lambda_j| + (1 - \mu)$ e il polinomio $p(\lambda) = q(\lambda)/q(1)$ che soddisfa $p(1) = 1$. Di conseguenza è stato provato

$$\begin{aligned} \|(I - \tilde{\mathscr{P}}_m)x_1\|_2 &= \min_{u \in \mathcal{K}_m(A, v_1^{\text{new}})} \|u - x_1\|_2 \\ &\leq \alpha^{k(m_1-1)} \left(\frac{\alpha + 1 - \mu}{2-\mu} \right)^k \cdot \xi \min_{p \in \mathcal{L}_{m-1}, p(\lambda_1)=1} \max_{\lambda \in Sp(A)/\lambda_1} |p(\lambda)|. \end{aligned}$$

dove la prima uguaglianza segue ancora dal Lemma 2 e si è usato il fatto che al variare di u la quantità $\frac{u}{\eta\gamma_1 q(1)(2-\mu)^k}$ descrive tutti i vettori del sottospazio $\mathcal{K}_m(A, v_1^{\text{new}})$. □

Si osserva che il fattore di miglioramento della convergenza è $\alpha^{k(m_1-1)} \left(\frac{\alpha+1-\mu}{2-\mu}\right)^k$.
Si può notare come questo fattore converga ad 1 per $\alpha \to 1$, quindi è ragionevole aspettarsi che il passaggio dal metodo PET al thick restarted Arnoldi sia più stabile per valori di α piccoli. Infatti all'aumentare di α il suddetto passaggio non comporterà particolari aumenti della velocità, ma l'accelerazione data dal thick restarted Arnoldi risulterà più evidente, in quanto il metodo PET avrebbe convergenza molto lenta.

Sperimentazioni numeriche

Tutte le sperimentazioni che seguono sono state condotte usando MATLAB R2020b sul sistema operativo macOS Catalina 10.15.7 con processore 2,3 GHz Quad-Core Intel Core i7 e memoria RAM 16 GB 1600 MHz DDR3.

Inizialmente si sono voluti replicare i risultati presenti nell'articolo di riferimento. Sono stati quindi considerati quattro set di dati distinti, riportati in Tabella 1, ed è stato effettuato un confronto tra i diversi metodi implementati sulla base del numero di iterazioni (It), del numero di prodotti matrice-vettore effettuati (Mv) e del tempo impiegato per raggiungere la tolleranza desiderata (T).
Nella tabella n è la dimensione della matrice, l il numero di dangling nodes, d la densità, e infine m, p, $maxit$ e m_1 sono i parametri descritti nelle sezioni precedenti. Inoltre alcune quantità sono state fissate universalmente: viene usato lo stesso vettore iniziale $x_0 = v = e/n$, la tolleranza è $tol = 10^{-8}$ e, come già anticipato, si usa il parametro $\beta = \alpha - 0.1$.
Queste sperimentazioni iniziali sono state condotte al variare di $\alpha = 0.99, 0.993, 0.995, 0.997$ sulla base di quelle effettuate nell'articolo di riferimento. Si aggiungono inoltre le sperimentazioni per $\alpha = 0.85, 0.90$ per osservare il comportamento dell'algoritmo per valori più bassi, maggiormente utilizzati nelle applicazioni reali del PageRank.

Nome	n	l	d	m	p	maxit	m_1
wb-cs-stanford	9914	2861	0.375×10^{-1}	5	3	8	50
web-Stanford	281903	172	0.291×10^{-1}	5	3	12	40
Stanford_Berkeley	683446	68062	0.162×10^{-2}	8	5	6	40
web-Google	916428	176974	0.608×10^{-3}	9	4	10	40

Tabella 1: Matrici di esempio con relativi set di dati

Oltre ai metodi PET e Arnoldi-PET (A-PET) descritti precedentemente, è stato poi implementato anche il metodo Power-Arnoldi (P-A), usato nell'articolo per il confronto.

Metodo Power-Arnoldi Analogo del metodo Arnoldi-PET, che utilizza il metodo delle potenze in sostituzione del metodo PET [2]. La differenza principale è che il Power-Arnoldi, invece di iniziare il procedimento con l'algoritmo thick restarted Arnoldi, prevede un'esecuzione del metodo delle potenze per ottenere un'approssimazione, quindi prosegue come nel metodo Arnoldi-PET. Sfrutta gli stessi parametri, con lo stesso significato, per il passaggio da un metodo all'altro.
L'algoritmo è presentato in Figura 12.

Inoltre, per completezza, è stato impiegato per il confronto anche l'algoritmo thick restarted Arnoldi (TRA), in modo da poter osservare il miglioramento della convergenza del metodo Arnoldi-PET rispetto a entrambi i metodi da cui è derivato.

Infine sono stati implementati il metodo delle potenze (P), presentato in Figura 8, e un algoritmo basato sull'iterazione di Arnoldi (AR), presentato in Figura 10, che utilizza una tecnica di restart più semplice basata sulla decomposizione in valori singolari (SVD) della matrice H_m [3].

Nelle Tabelle 2-5 e nelle Figure 1-4 sono riportati rispettivamente i dati e i plot delle norme dei residui in funzione delle iterazioni relativi alle sperimentazioni effettuate.

Dai grafici, anzitutto, è possibile notare come, rispetto al metodo PET, con il metodo Arnoldi-PET si abbassi il numero di iterazioni in rapporto alla discesa dell'errore. In generale di conseguenza risulta anche migliore del metodo delle potenze.

Ponendo l'attenzione sui due metodi ibridi, si nota come per la matrice *wb-cs-stanford* (Tabella 2) risulti sempre migliore il metodo Power-Arnoldi, mentre per le altre si abbia in media il comportamento opposto, in particolare per α più elevato.
Si può supporre che il metodo Power-Arnoldi sia più veloce in generale per α piccoli in quanto, a differenza dell'Arnoldi-PET, l'algoritmo inizia con l'iterazione del metodo delle potenze, che quindi risulta più efficace in poche iterazioni.
È possibile invece dedurre che la differenza tra le matrici risieda nel modo in cui il metodo delle potenze e il metodo PET influiscono sul comportamento del thick restarted Arnoldi ad ogni iterazione. Si è visto nel Teorema 4 come il valore m_1 influisca sul metodo Arnoldi. Nelle sperimentazioni successive si vedrà in effetti che per distinti valori del parametro il metodo risulterà velocizzato anche per la prima matrice.

Si osserva inoltre come i due metodi ibridi risultino più veloci rispetto al metodo delle potenze e del metodo PET per le due matrici di dimensioni minori, mentre siano più lenti altrimenti. In alcuni casi, in associazione a questo, si ha che, rispetto ai due metodi base, si hanno un maggior numero di prodotti matrici-vettore.
Si può supporre che questo segua da una implementazione non ottimale del metodo thick restarted Arnoldi, e dalla particolare scelta del vettore iniziale da usare una volta completate le iterazione nell'algoritmo basato sull'iterazione di Arnoldi. Quest'ultimo non era descritto precisamente nell'articolo di riferimento [1], e nell'articolo da cui è stato studiato il metodo Power-Arnoldi [2], di conseguenza la scelta operata potrebbe non essere ideale.
Tale scelta potrebbe allo stesso modo essere il motivo del comportamento anomalo che si può notare nel caso della matrice *Stanford-Berkeley* per $\alpha = 0.997$, si veda la Figura 3.

In generale è comunque possibile confermare il miglioramento del metodo Arnoldi-PET rispetto al metodo Power-Arnoldi, in particolare in caso di valori alti di α. Il peggioramento dei due metodi rispetto ai metodi base si può spiegare con un'implementazione non ottimale dei metodi dovuta alla mancanza di indicazioni specifiche negli articoli di riferimento, ma è comunque possibile osservare un netto miglioramento rispetto all'algoritmo thick restarted Arnoldi, come voluto.

	α	P	PET	AR	TRA	P-A	A-PET
It		65	61	13	25	32	38
Mv	0.85	65	61	65	52	49	54
T		0.100	0.020	0.141	0.076	0.050	0.032
It		97	87	17	35	43	56
Mv	0.90	97	87	85	72	68	79
T		0.028	0.024	0.061	0.052	0.051	0.046
It		998	650	88	324	166	168
Mv	0.99	998	650	440	649	250	275
T		0.240	0.146	0.251	0.427	0.097	0.113
It		1428	900	119	422	191	199
Mv	0.993	1428	900	595	845	291	312
T		0.288	0.202	0.326	0.610	0.101	0.106
It		2001	1150	196	530	214	230
Mv	0.995	2001	1150	980	1061	329	359
T		0.386	0.224	0.479	0.806	0.122	0.132
It		3338	1650	206	717	247	334
Mv	0.997	3338	1650	1030	1435	391	534
T		0.636	0.343	0.517	0.998	0.135	0.183

Tabella 2: Dati per la matrice wb-cs-stanford

	α	P	PET	AR	TRA	P-A	A-PET
It		72	62	18	34	53	39
Mv	0.85	72	62	90	72	62	56
T		0.371	0.338	1.079	1.303	0.423	0.470
It		110	94	25	50	86	53
Mv	0.90	110	94	125	102	95	79
T		0.694	0.510	1.504	1.870	0.587	0.672
It		1141	680	168	352	322	275
Mv	0.99	1141	680	840	705	468	381
T		6.041	3.719	10.154	13.087	4.160	3.099
It		1632	880	229	558	409	319
Mv	0.993	1632	880	1145	1116	569	433
T		8.324	4.507	13.531	20.430	4.759	3.599
It		2287	1200	335	716	363	327
Mv	0.995	2287	1200	1675	1433	505	458
T		11.924	6.142	19.871	26.070	4.049	3.751
It		3815	1760	498	1066	526	423
Mv	0.997	3815	1760	2490	2133	738	596
T		19.585	9.046	29.479	38.974	5.878	4.807

Tabella 3: Dati per la matrice web-Stanford

	α	P	PET	AR	TRA	P-A	A-PET
It		70	56	11	24	29	33
Mv	0.85	70	56	88	73	57	60
T		0.917	0.742	2.885	3.459	1.418	1.439
It		107	80	16	35	52	46
Mv	0.90	107	80	128	106	80	85
T		1.355	1.019	3.793	5.058	1.711	2.041
It		1071	600	118	250	221	217
Mv	0.99	1071	600	944	754	454	429
T		14.060	7.619	27.688	34.786	11.364	10.619
It		1513	800	169	350	304	279
Mv	0.993	1513	800	1352	1054	620	594
T		21.312	10.168	40.285	45.402	15.609	15.175
It		2086	1080	230	471	433	302
Mv	0.995	2086	1080	1840	1417	942	627
T		29.653	13.742	54.551	64.956	24.362	16.001
It		3368	1680	359	732	566	1026
Mv	0.997	3368	1680	2872	2200	1336	2364
T		48.481	21.324	85.345	97.550	35.552	63.263

Tabella 4: Dati per la matrice Stanford_Berkeley

	α	P	PET	AR	TRA	P-A	A-PET
It		68	59	9	14	29	27
Mv	0.85	68	59	81	78	67	64
T		1.869	1.472	4.327	5.509	3.161	3.109
It		104	87	14	23	41	39
Mv	0.90	104	87	126	123	96	76
T		2.526	2.080	6.247	8.165	4.071	3.222
It		1085	600	121	226	348	279
Mv	0.99	1085	600	1089	1138	854	676
T		24.227	13.640	58.852	77.598	38.225	29.127
It		1553	800	162	312	437	399
Mv	0.993	1553	800	1458	1568	1086	959
T		35.673	18.463	80.070	105.732	50.318	43.311
It		2176	1000	213	427	606	559
Mv	0.995	2176	1000	1917	2143	1506	1353
T		48.619	22.921	102.767	144.088	70.008	59.146
It		3630	1400	317	690	928	799
Mv	0.997	3630	1400	2853	3458	2296	1955
T		81.262	31.998	154.576	213.462	105.232	89.154

Tabella 5: Dati per la matrice web-Google

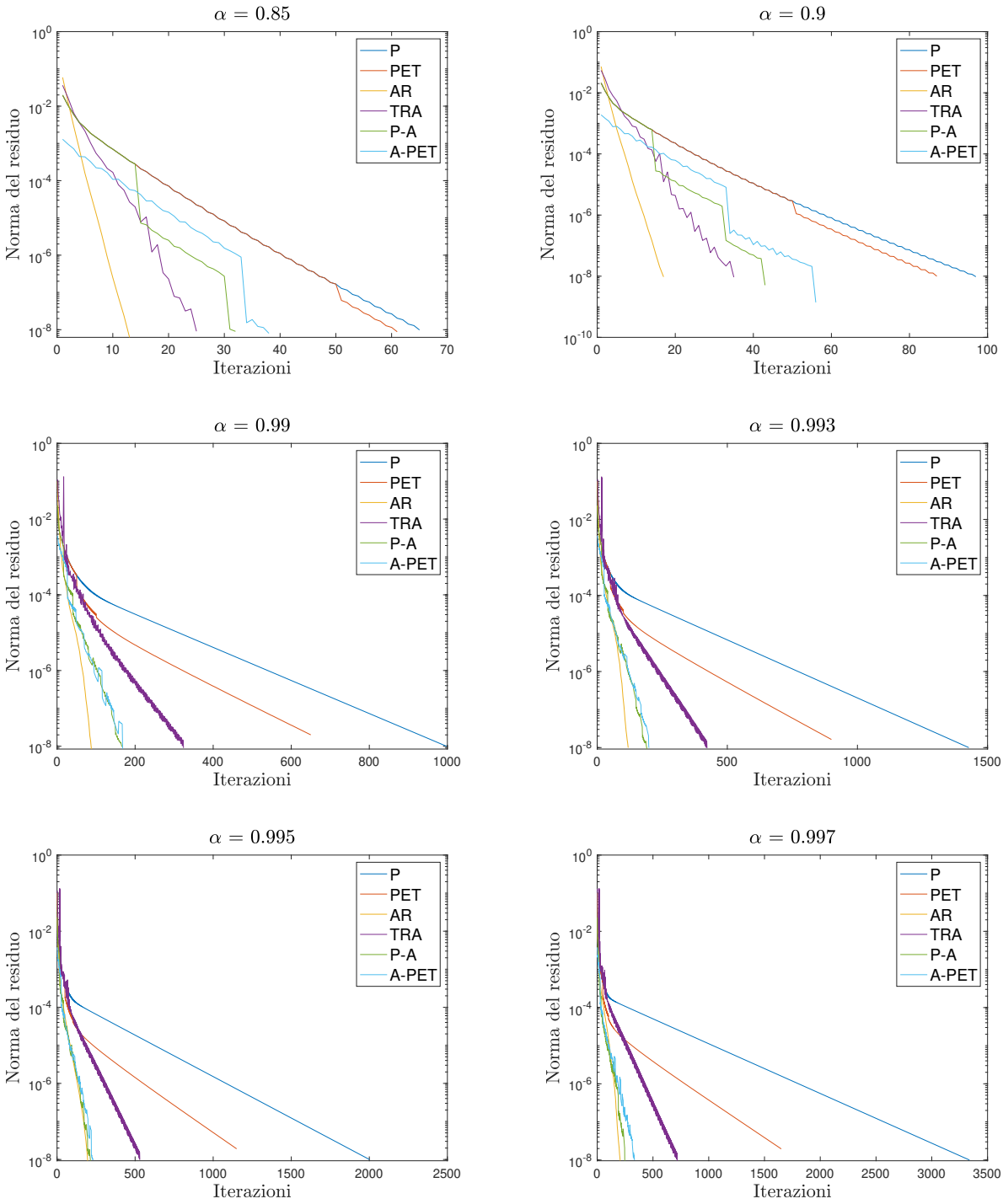

Figura 1: Plot per la matrice wb-cs-stanford

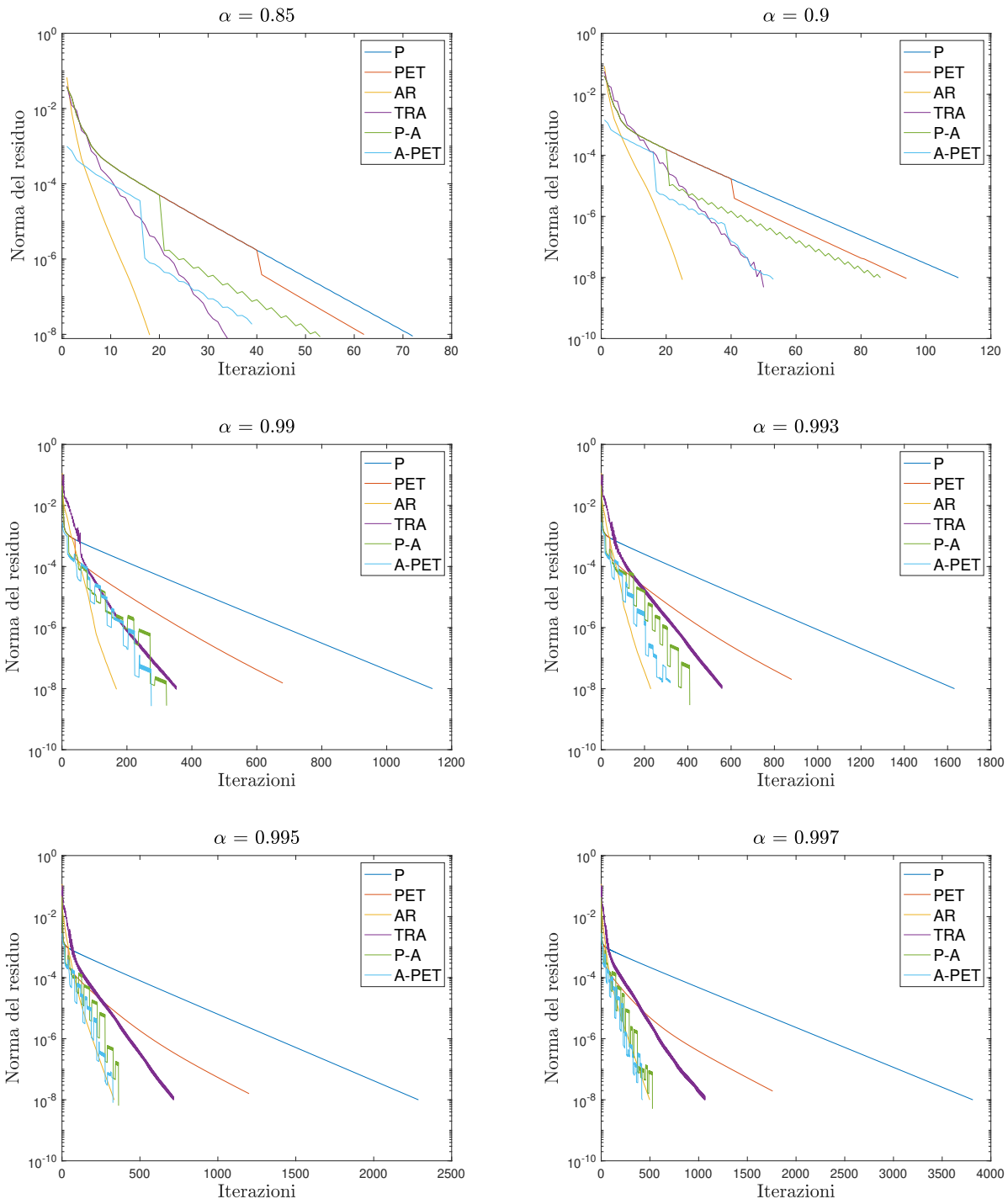

Figura 2: Plot per la matrice web-Stanford

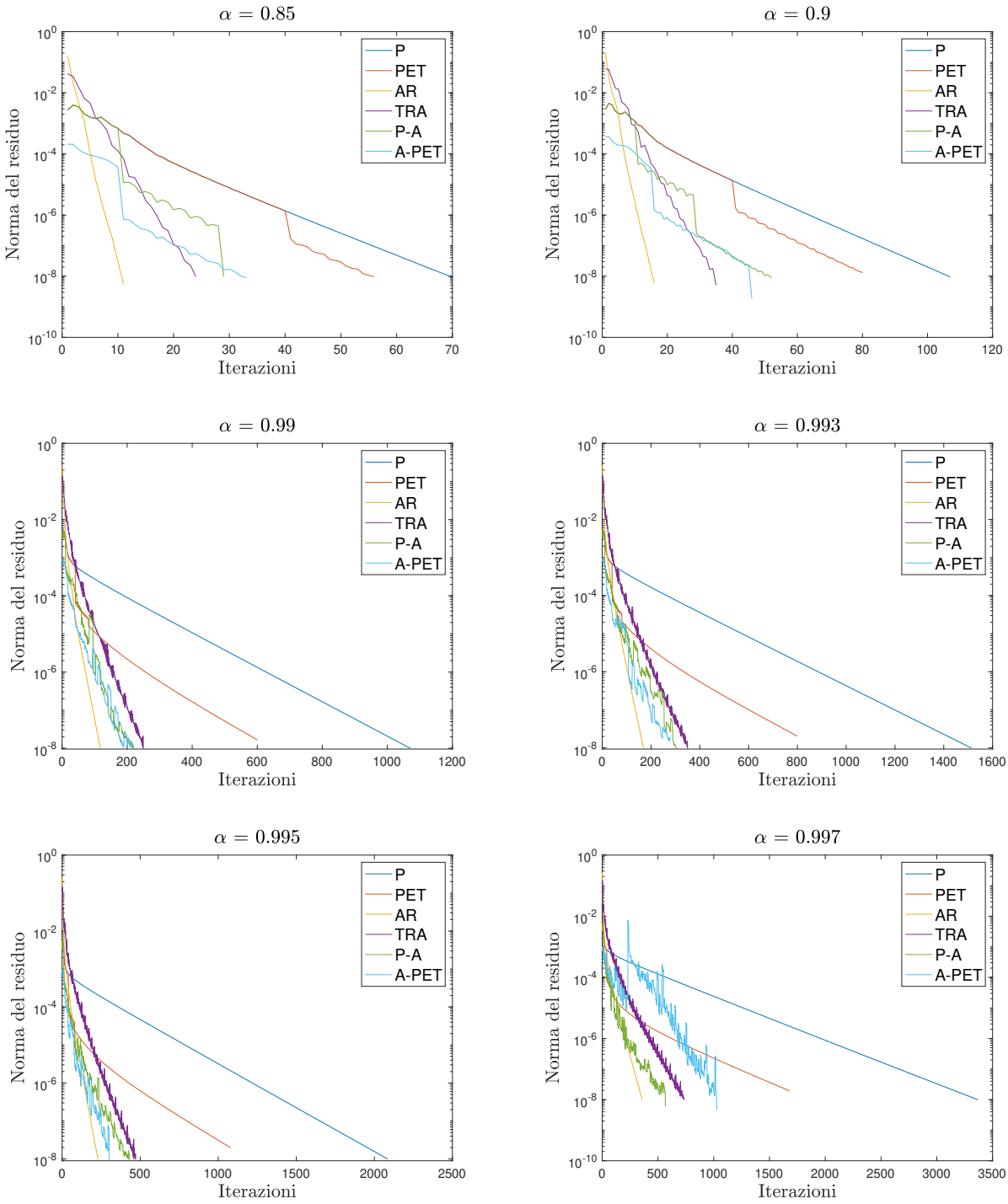

Figura 3: Plot per la matrice Stanford_Berkeley

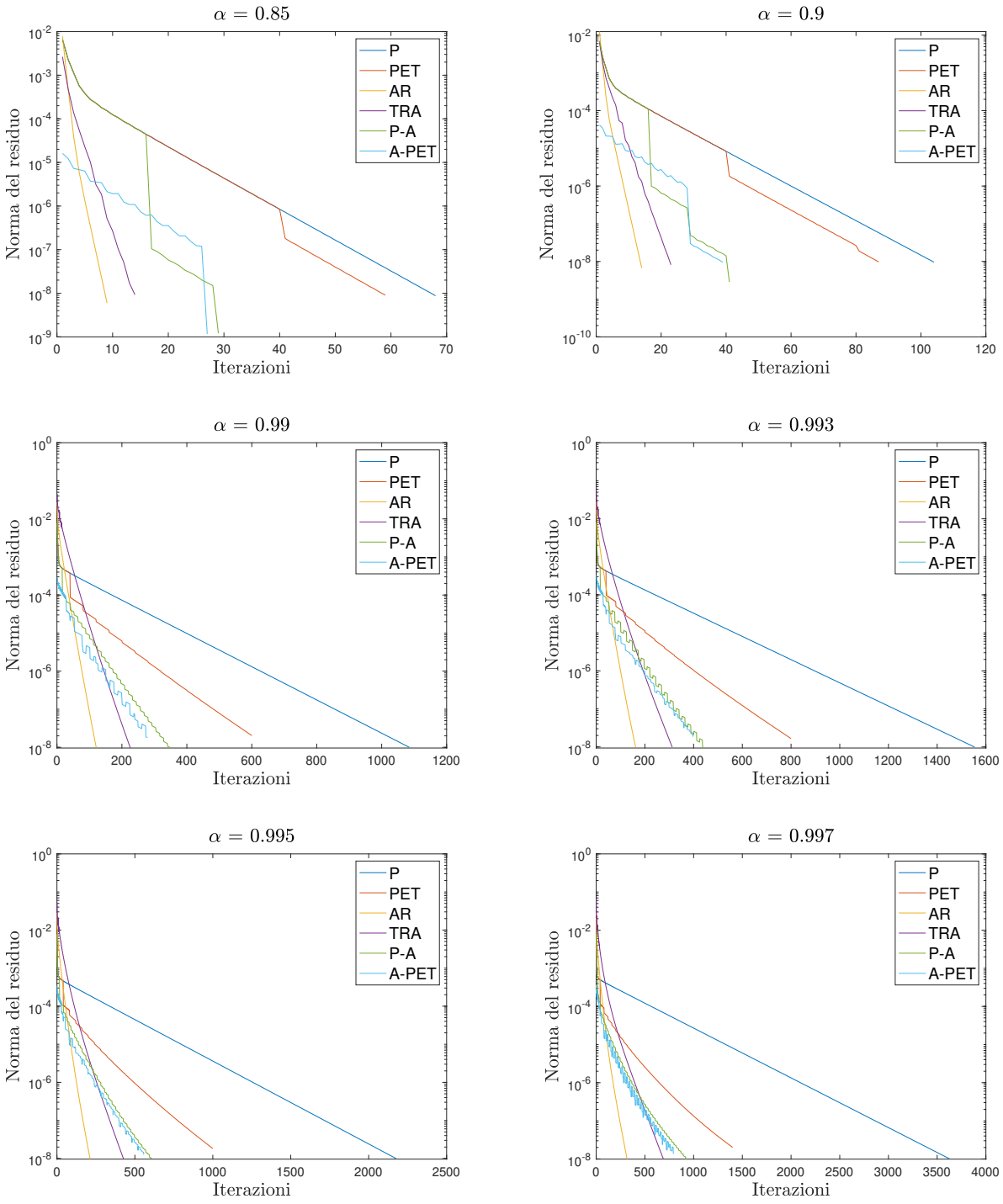

Figura 4: Plot per la matrice web-Google

Avendo considerato come ascisse il numero di iterazioni, i metodi che si basano unicamente su Arnoldi appaiono velocizzati guardando esclusivamente ai grafici, in quanto per ogni iterazione si ha più di un prodotto matrice-vettore. Questo si evidenzia nei tempi di esecuzione, che sono nettamente più alti in confronto agli altri metodi.
Si mostra quindi come esempio nella Figura 5 il grafico per la matrice web-Stanford con $\alpha = 0.99$ dove si usano come ascisse il numero di prodotti matrice-vettore, e come ordinate ancora le norme dei residui.

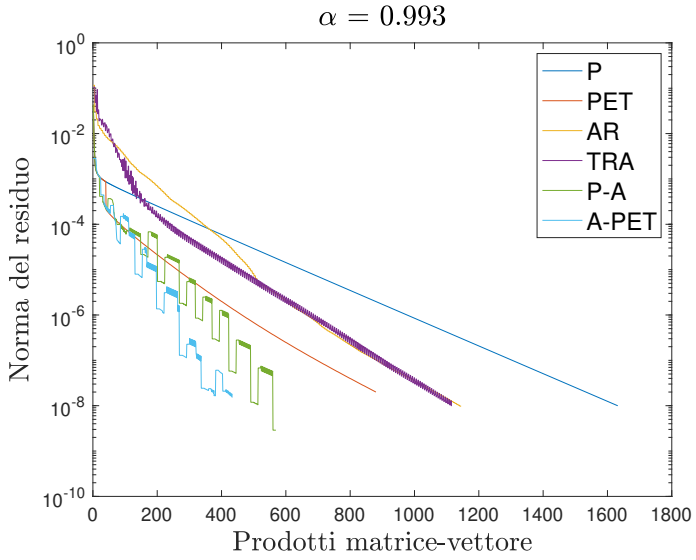

Figura 5: Plot della norma del residuo in funzione del numero di prodotti matrice-vettore per la matrice web-Stanford.

Si nota in realtà come il metodo Arnoldi con restart basato sulla SVD risulti in media più veloce del thick restarted Arnoldi, in particolare per valori alti del damping factor.

Sulla base di questa osservazione si è condotta la seguente sperimentazione: si è considerato il metodo Arnoldi-PET e si è utilizzato il metodo Arnoldi basato sulla SVD in sostituzione dell'algoritmo thick restarted Arnoldi, mantenendo inalterati i parametri utilizzati per il passaggio da una sezione all'altra della procedura.
Come nel metodo Arnoldi-PET si eseguono solo due iterazioni dell'algoritmo basato su Arnoldi. Ne risulta quindi il codice presentato in Figura 14 (A-PET-T).

Esso è sostenuto dalle stesse basi teoriche valide per l'algoritmo Arnoldi-PET. In effetti il risultato sul miglioramento della convergenza visto nel Teorema 4 vale per l'iterazione di Arnoldi, e non per lo specifico algoritmo thick restarted Arnoldi; di conseguenza è applicabile anche in questo caso.

È stato quindi confrontato il nuovo algoritmo con i due metodi Power-Arnoldi e Arnoldi-PET. La sperimentazione è stata condotta per i valori di $\alpha = 0.90, 0.995$ per osservare l'andamento dell'errore e i diversi comportamenti per un valore alto e uno basso del damping factor.
Sono state prese in considerazione la matrice web-Stanford, che ha, tra tutte quelle prese in analisi, il minimo numero di dangling nodes in proporzione alla taglia, e la matrice web-Google, che è la matrice di taglia maggiore e di densità minore.

I risultati sono riportati nelle Tabelle 6-7 e nelle Figure 6-7.

In entrambi i casi, per $\alpha = 0.995$, si nota un miglioramento del metodo rispetto ai precedenti. In particolare quest'ultimo risulta più stabile nell'alternanza tra un metodo e l'altro, in quanto i precedenti hanno parziali innalzamenti del residuo dovuti alla scelta del vettore iniziale nel passaggio dall'algoritmo thick restarted Arnoldi ai rispettivi metodi specifici.

È possibile quindi confermare l'ipotesi che l'implementazione dei due metodi sia stata penalizzata dalla mancanza di informazioni specifiche riguardo alla scelta di tale vettore.

	α	P-A	A-PET	A-PET-T
It		86	53	46
Mv	0.90	95	79	86
T		0.587	0.672	0.72699
It		363	327	257
Mv	0.995	505	458	437
T		4.049	3.751	3.5082

Tabella 6: Dati per la matrice web-Stanford

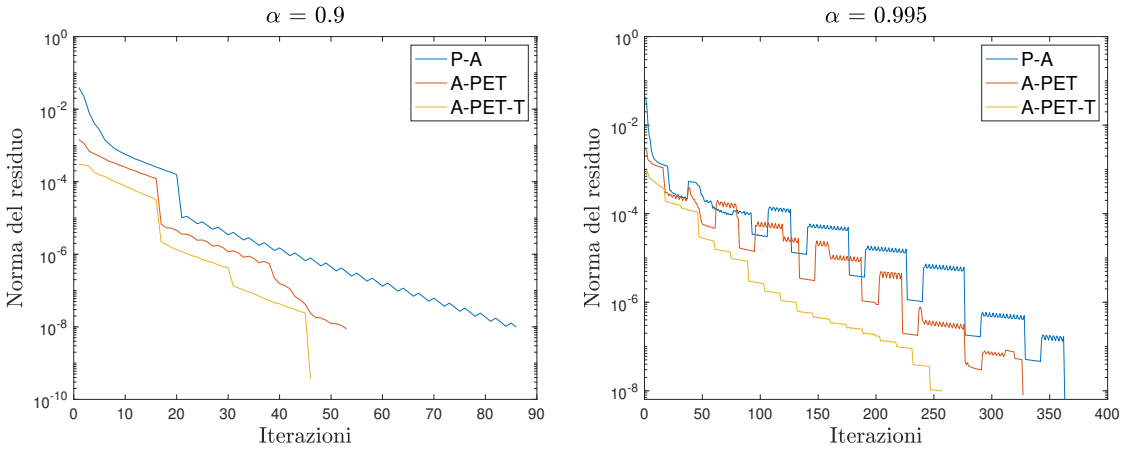

Figura 6: Plot per la matrice web-Stanford

	α	P-A	A-PET	A-PET-T
It		41	39	25
Mv	0.90	96	76	61
T		4.071	3.222	2.5811
It		606	559	359
Mv	0.995	1506	1353	899
T		70.008	59.146	37.0996

Tabella 7: Dati per la matrice web-Google

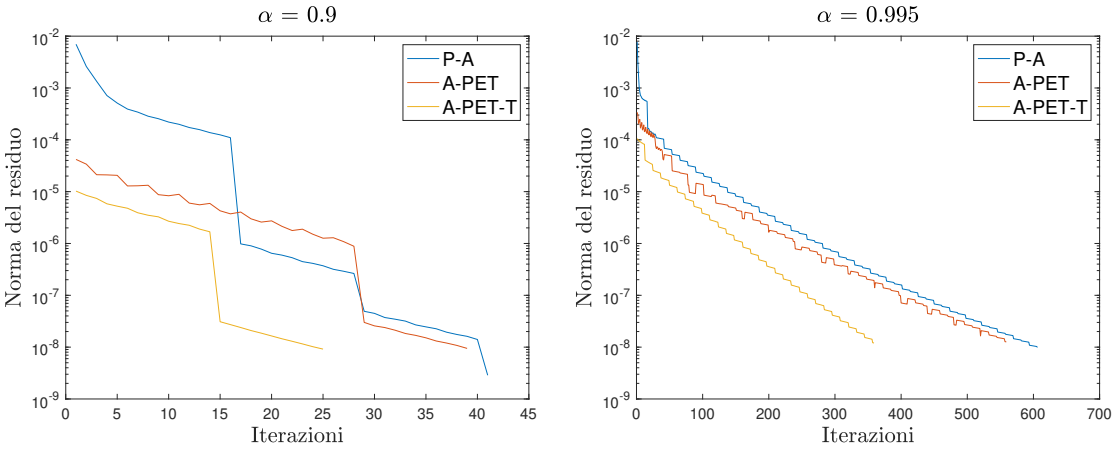

Figura 7: Plot per la matrice web-Google

Si mostra infine, come ultima sperimentazione, il miglioramento del metodo Arnoldi-PET rispetto al Power-Arnoldi anche per la matrice *wb-cs-stanford* nel caso di una scelta del parametro m_1 più oculata, come anticipato.
Si paragonano quindi i valori trovati per il metodo Power-Arnoldi, che non è influenzato da tale parametro, con i nuovi valori individuati per il metodo Arnoldi-PET ponendo $m_1 = 95$.
I risultati sono mostrati in Tabella 8, dove sono stati riportati i valori di α per cui il nuovo parametro migliora la convergenza del metodo Arnoldi-PET.

	α	P-A	A-PET
It		191	189
Mv	0.993	291	286
T		0.101	0.088
It		247	235
Mv	0.997	391	374
T		0.135	0.126

Tabella 8: Dati per la matrice wb-cs-stanford con $m_1 = 95$

Codice MATLAB

Figura 8: Metodo delle potenze

```
function [x,err,mv,t] = power_method(H,alpha,v,x,tol)
% function [x,err,mv,t] = power_method(H,alpha,v,x,tol)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         x: unit positive initial
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

% Size of matrix.
n = size (H,1);

% Matrix D.
e = ones (n,1);
d = H * e;
dang = d==0;
dh = d + dang*n;
dh = 1./dh;

% Set r = 1 and k = 0.
r = 1;
k = 0;

% Errors and matrix-vector products.
err = [];
mv = 0;

tic;

while r > tol

    % xn = A * x;
    xn = x .* dh;
    xn = H' * xn + sum(dang .* xn);
    xn = xn * alpha + (1 - alpha) * v;
    mv = mv + 1;

    k = k + 1;

    r = norm(xn - x,2);
    err = [err r];%#ok<AGROW>
    xn = xn / norm(xn,1);
    if r <= tol
        break;
    end

    x = xn;
end

t = toc;
```

Figura 9: Metodo PET

```
function [x,err,mv,t] = pet_method(H,alpha,v,x,m1,tol)
% function [x,err,mv,t] = pet_method(H,alpha,v,x,m1,tol)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         x: unit positive initial
%         m1: frequency of using the extrapolation based on trace
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

% Size of matrix.
n = size (H,1);

% Matrix D.
e = ones (n,1);
d = H * e;
dang = d==0;
dh = d + dang*n;
dh = 1./dh;

% Set r = 1 and k = 0.
r = 1;
k = 0;

% Compute the number of dangling nodes l and the trace mu.
l = sum(dang);
mu = 1 + alpha * (l / n - 1);

% Errors and matrix-vector products.
err = [];
mv = 0;

tic;

xn = x;
while r > tol

    % Run the power iteration m_1 steps to obtain x_{m_1-1} and x_{m_1}.
    for i = 1:m1
        x = xn;

        % xn = A * x;
        xn = x .* dh;
        xn = H' * xn + sum(dang .* xn);
        xn = xn * alpha + (1 - alpha) * v;
        mv = mv + 1;

        k = k + 1;

        r = norm(xn - x,2);
        err = [err r];%#ok<AGROW>
        xn = xn / norm(xn,1);
        if r <= tol
            break;
        end
    end
```

```
    if r <= tol
        x = xn;
        break;
    end

    % Use the extrapolation scheme based on x_{m_1-1}, x_{m_1} and mu.
    x = xn - (mu - 1) * x;
    x = x / norm(x,1);
    r = norm(x - xn,2);
    xn = x;
end

t = toc;
```

Figura 10: Metodo Arnoldi con restart

```
function [x,err,mv,t] = arnoldi_type_algorithm(H,alpha,v,v1,m,tol)
% function [x,err,mv,t] = arnoldi_type_algorithm(H,alpha,v,v1,m,tol)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         v1: unit positive initial
%         m: dimension of the Krylov subspace
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

% Errors and matrix-vector products.
err = [];
r = 1;
mv = 0;

tic;

while r > tol

    % Apply Algorithm 2 to form Vm1 = V_{m+1}, H_m, Hb = \bar{H_m}.
    [Hb,Vm1] = arnoldi_process(H,alpha,v,v1,m);
    mv = mv + m;
    Vm = Vm1(:,1:m);

    Hb = Hb - [eye(m);zeros(1,m)];
    [~,~,V] = svd(Hb);
    v1 = Vm * V(:,m);

    r = svds(Hb,1,'smallestnz');
    err = [err r];%#ok<AGROW>
end

t = toc;

x = sign(sum(v1))*v1;
x = x/norm(x,1);
```

Figura 11: Metodo thick restarted Arnoldi

```
function [x,err,mv,t] = thick_restarted_arnoldi(H,alpha,v,v1,m,p,tol)
% function [x,err,mv,t] = thick_restarted_arnoldi(H,alpha,v,v1,m,p,tol)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         v1: unit positive initial
%         m: dimension of the Krylov subspace
%         p: number of approximate eigenpairs which are wanted
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

em = zeros(m,1);
em(m) = 1;
em1 = [0;em];

% Errors and matrix-vector products.
err = [];
r = 1;
mv = 0;

tic;

% Apply Algorithm 2 to form Vm1 = V_{m+1}, H_m, Hb = \bar{H_m}.
[Hb,Vm1] = thick_arnoldi_process(H,alpha,v,v1,[],m);
mv = mv + m;
Hm = Hb(1:m,:);

% Compute all the eigenpairs (li, yi) (i = 1, 2, ... , m) of the matrix Hm.
[Y,L] = eig(Hm);
L = diag(L);

% Then select p largest of them and turn to step 5.
[~,I] = sort(abs(L),'descend');
Y = Y(:,I);
Y = Y(:,1:p);

% Check convergence. If the largest eigenpairs is accurate enough, i.e.,
% hm+1,m|e_m^T y1|<= tol, then take x1 = Vm y1 as an approximation to the
% PageRank vector and stop, else continue (only after first iteration).
while r > tol

    Wp = zeros(m,p-1);
    p1 = 1;
    i = 1;
    while i <= p
        if sum(round(real(Y(:,i)),10) ~= zeros(m,1))
            Wp(:,p1) = real(Y(:,i));
            p1 = p1 + 1;
        end
        if sum(round(imag(Y(:,i)),10) ~= zeros(m,1))
            Wp(:,p1) = imag(Y(:,i));
            p1 = p1 + 1;
            i = i + 1;
        end
```

```
        if p1 > p
            break;
        end
        i = i + 1;
    end

    % Orthonormalize yi (i = 1,2,...,p) to form a real m x p matrix.
    Wp = gramschmidt(Wp);
    p1 = size(Wp,2);

    % By appending a zeros row at the bottom of the matrix Wp, form a real
    % (m + 1) x p matrix Wpt = [Wp;0], and set
    % Wp+1 = [Wpt , em+1 ], where em+1 is the (m + 1)th co-ordinate vector. Note
    % that Wp+1 is an (m + 1) x (p + 1) orthonormal matrix.
    Wpt = [Wp;zeros(1,p1)];
    Wp1 = [Wpt,em1];

    % Use the old Vm+1 and Hm to form portions of the new Vm+1 and Hm. Let
    % Vp+1 = Vm+1Wp+1, \bar{Hp} = Wp+1^T\bar{Hm}Wp, return to step 3.
    Vp1 = Vm1 * Wp1;
    Hpb = Wp1' * Hb * Wp;

    % Apply the Arnoldi process from the current point vp+1 to form
    % Vm+1,Hm, \bar{Hm}. Compute all the eigenpairs (li,yi) (i = 1, 2, ... , m)
    % of the matrix Hm. Then select p largest of them.
    [Hb,Vm1] = thick_arnoldi_process(H,alpha,v,Vp1,Hpb,m);
    mv = mv + m - p1;
    Hm = Hb(1:m,:);
    [Y,L] = eig(Hm);
    L = diag(L);
    [~,I] = sort(abs(L),'descend');
    Y = Y(:,I);
    Y = Y(:,1:p);

    r = Hb(m+1,m)*abs(em'* Y(:,1));
    err = [err r];%#ok<AGROW>
end

t = toc;

x = Vm1(:,1:m) * Y(:,1);
x = sign(sum(x))*x;
x = x/norm(x,1);
```

Figura 12: Metodo Power-Arnoldi

```
function [x,err,mv,t] = power_arnoldi(H,alpha,v,x,m,beta,maxit,p,tol)
% function [x,err,mv,t] = power_arnoldi(H,alpha,v,x,m,beta,maxit,p,tol);
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         x: unit positive initial
%         m: dimension of the Krylov subspace
%         beta, maxit: parameters used to flip-flop between the PET
%                      method and the thick restarted Arnoldi algorithm
%         p: number of approximate eigenpairs which are wanted
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

% Size of matrix.
n = size (H,1);

% Dangling nodes.
e = ones (n,1);
d = H * e;
dang = d==0;
dh = d + dang*n;
dh = 1./dh;

% Set r = 1, r0 = r and k = 1.
r = 1;
r0 = r;
k = 1;

% Errors and matrix-vector products.
err = [];
mv = 0;

% First run of thick restarted Arnoldi.
fr = 1;

% Parameter used to flip-flop between the PET method and the thick
% restarted Arnoldi algorithm.
restart = 0;

tic;

% Run the power method once.
xn = x;
while r > tol && restart < maxit
    x = xn;

    % xn = A * x;
    xn = x .* dh;
    xn = H' * xn + sum(dang .* xn);
    xn = xn * alpha + (1 - alpha) * v;
    mv = mv + 1;

    k = k + 1;
    r = norm(xn - x,2);
    xn = xn / norm(xn,1);
    err = [err r]; %#ok<AGROW>
```

```
60      if r < tol
61          break;
62      end
63      if r / r0 > beta
64          restart = restart + 1;
65      end
66      r0 = r;
67  end
68
69  x = xn;
70
71  while r > tol
72
73      % Run 2 iterations of thick restarted arnoldi. Iterate
74      % steps 2-8 of Algorithm 4 for the first run and steps 3-8 otherwise.
75      if fr == 1
76          [x,x1t,r,mv] = short_tra_fr(H,alpha,v,x,m,p,tol,mv);
77          fr = 0;
78      else
79          [x,x1t,r,mv] = short_tra(H,alpha,v,x,m,p,tol,mv);
80      end
81
82      % If the residual norm satisfies the prescribed tolerance tol, then
83      % stop, else continue.
84      if r <= tol
85          break;
86      end
87
88      % Run the power method with x1t as the initial guess, where x1t is the
89      % approximation vector obtained from the thick restarted Arnoldi
90      % algorithm.
91      x1t = abs(x1t);
92      xn = x1t / norm (x1t,1);
93      restart = 0;
94
95      while r > tol && restart < maxit
96          ratio = 0;
97          r0 = r;
98          r1 = r;
99          while r > tol && ratio < beta
100             x = xn;
101
102             % xn = A * x;
103             xn = x .* dh;
104             xn = H' * xn + sum(dang .* xn);
105             xn = xn * alpha + (1 - alpha) * v;
106             mv = mv + 1;
107
108             r = norm(xn - x,2);
109             xn = xn / norm(xn,1);
110             ratio = r / r0;
111             r0 = r;
112             k = k + 1;
113             err = [err r];%#ok<AGROW>
114         end
115         x = xn;
116         if r / r1 > beta
117            restart = restart + 1;
118         end
119     end
120 end
121
122 t = toc;
```

Figura 13: Metodo Arnoldi-PET

```
function [x,err,mv,t] = arnoldi_pet(H,alpha,v,x,m,m1,beta,maxit,p,tol)
% function [x,err,mv,t] = arnoldi_pet(H,alpha,v,x,m,m1,beta,maxit,p,tol)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         x: unit positive initial
%         m: dimension of the Krylov subspace
%         m1: frequency of using the extrapolation based on trace
%         beta, maxit: parameters used to flip-flop between the PET
%                      method and the thick restarted Arnoldi algorithm
%         p: number of approximate eigenpairs which are wanted
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

% Size of matrix.
n = size (H,1);

% Dangling nodes.
e = ones (n,1);
d = H * e;
dang = d==0;
dh = d + dang*n;
dh = 1./dh;

% Set r = 1 and k = 1.
r = 1;
k = 1;

% Errors and matrix-vector products.
err = [];
mv = 0;

% First run of thick restarted Arnoldi.
fr = 1;

% Compute the number of dangling nodes l and the trace mu.
l = sum(dang);
mu = 1 + alpha * (l / n - 1);

tic;

while r > tol

    % Run 2 iterations of thick restarted arnoldi. Iterate
    % steps 2-7 of Algorithm 3 for the first run and steps 3-7 otherwise.
    if fr == 1
        [x,x1t,r,mv] = short_tra_fr(H,alpha,v,x,m,p,tol,mv);
        fr = 0;
    else
        [x,x1t,r,mv] = short_tra(H,alpha,v,x,m,p,tol,mv);
    end
```

```
    % If the residual norm satisfies the prescribed tolerance tol, then
    % stop, else continue.
    if r <= tol
        break;
    end

    % Run the PET method with x1t as the initial guess, where x1t is the
    % approximation vector obtained from the thick restarted Arnoldi
    % algorithm.
    x1t = abs(x1t);
    xn = x1t / norm (x1t,1);
    restart = 0;

    while r > tol && restart < maxit
        ratio = 0;
        r0 = r;
        r1 = r;
        while r > tol && ratio < beta
            x = xn;

            % xn = A * x;
            xn = x .* dh;
            xn = H' * xn + sum(dang .* xn);
            xn = xn * alpha + (1 - alpha) * v;
            mv = mv + 1;

            r = norm(xn - x,2);
            xn = xn / norm(xn,1);
            ratio = r / r0;
            r0 = r;
            k = k + 1;
            err = [err r];%#ok<AGROW>

            if (mod(k,m1)==0)
                x = xn - (mu - 1) * x;
                x = x / norm(x,1);
                r = norm(x - xn,2);
                if r <= tol
                    break;
                end
                xn = x;
            end
        end
        x = xn;
        if r / r1 > beta
           restart = restart + 1;
        end
    end
end

t = toc;
```

Figura 14: Metodo Arnoldi-PET-Test

```
function [x,err,mv,t] = arnoldi_pet_test(H,alpha,v,x,m,m1,beta,maxit,tol)
% function [x,err,mv,t] = arnoldi_pet_test(H,alpha,v,x,m,m1,beta,maxit,tol)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         x: unit positive initial
%         m: dimension of the Krylov subspace
%         m1: frequency of using the extrapolation based on trace
%         beta, maxit: parameters used to flip-flop between the PET
%                      method and the Arnoldi type algorithm
%         tol: prescribed tolerance
% Output - x: PageRank vector
%          err: vector containing errors
%          mv: number of matrix-vector products
%          t: time elapsed

% Size of matrix.
n = size (H,1);

% Dangling nodes.
e = ones (n,1);
d = H * e;
dang = d==0;
dh = d + dang*n;
dh = 1./dh;

% Set r = 1 and k = 1.
r = 1;
k = 1;

% Errors and matrix-vector products.
err = [];
mv = 0;

% Compute the number of dangling nodes l and the trace mu.
l = sum(dang);
mu = 1 + alpha * (l / n - 1);

tic;

while r > tol

    % Run 2 iterations of Arnoldi type algorithm.
    [x,r,mv] = short_ar(H,alpha,v,x,m,tol,mv);

    % If the residual norm satisfies the prescribed tolerance tol, then
    % stop, else continue.
    if r <= tol
        break;
    end

    % Run the PET method with x as the initial guess, where x is the
    % approximation vector obtained from the Arnoldi type algorithm.
    xn = x;
    restart = 0;
```

```
56      while r > tol && restart < maxit
57          ratio = 0;
58          r0 = r;
59          r1 = r;
60          while r > tol && ratio < beta
61              x = xn;
62
63              % xn = A * x;
64              xn = x .* dh;
65              xn = H' * xn + sum(dang .* xn);
66              xn = xn * alpha + (1 - alpha) * v;
67              mv = mv + 1;
68
69              r = norm(xn - x,2);
70              xn = xn / norm(xn,1);
71              ratio = r / r0;
72              r0 = r;
73              k = k + 1;
74              err = [err r];%#ok<AGROW>
75
76              if (mod(k,m1)==0)
77                  x = xn - (mu - 1) * x;
78                  x = x / norm(x,1);
79                  r = norm(x - xn,2);
80                  if r <= tol
81                      break;
82                  end
83                  xn = x;
84              end
85          end
86          x = xn;
87          if r / r1 > beta
88             restart = restart + 1;
89          end
90      end
91  end
92
93  t = toc;
```

A Codici ausiliari

Figura 15: Iterazione di Arnoldi

```
function [Hb,V] = thick_arnoldi_process(H,alpha,v,Vp,Hp,m)
% function [Hb,V] = thick_arnoldi_process(H,alpha,v,Vp,Hp,m)
% Input - H: adjacency matrix
%         alpha: damping factor
%         v: personalization vector
%         Vp: first part of the output V
%         Hp: first part of the output Hb
%         m: dimension of the Krylov subspace
% Output - Hb: upper Hessenberg matrix
%          V: orthonormal basis of the Krylov subspace

% Size of matrix.
n = size (H,1);

% Calculate matrix D.
e = ones (n,1);
d = H * e;
dang = d==0;
dh = d + dang*n;
dh = 1./dh;

% Hb = \bar{H_m}, V = V_{m+1}.
Hb = zeros(m+1 , m);
Hb(1:size(Hp,1),1:size(Hp,2))= Hp;
V = zeros(n , m+1);
V(1:size(Vp,1),1:size(Vp,2))= Vp;

k = size(Vp,2);

for j = k : m

    % q = A * V(:,j);
    q = V(:,j) .* dh;
    q = H' * q + sum(dang .* q);
    q = q * alpha + (1 - alpha) * v * sum(V(:,j));

    for i = 1 : j
        Hb(i,j) = V(:,i)' * q;
        q = q - Hb(i,j) * V(:,i);
    end

    Hb(j+1,j) = norm(q,2);

    if Hb(j+1,j) == 0
        break;
    end

    V(:,j+1) = q / Hb(j+1,j);
end
```

Figura 16: Ortogonalizzazione di Gram Schmidt

```
function U = gramschmidt(V)
% function U = gramschmidt(V)
% Input - V: generic matrix
% Output - U: orthonormal basis for the range of V

[n,k] = size(V);
U = zeros(n,k);
U(:,1) = V(:,1)/norm(V(:,1));
for i = 2:k
    U(:,i)=V(:,i);
    for j=1:i-1
        U(:,i)=U(:,i)-(U(:,j)'*U(:,i) )/(norm(U(:,j)))^2 * U(:,j);
    end
    U(:,i) = U(:,i)/norm(U(:,i));
end
```

Riferimenti bibliografici

[1] Qian-Ying Hu, Chun Wen, Ting-Zhu Huang, Zhao-Li Shen, Xian-Ming Gu, *A variant of the Power–Arnoldi algorithm for computing PageRank*, Journal of Computational and Applied Mathematics, Volume 381, 2021, 113034, ISSN 0377-0427, https://doi.org/10.1016/j.cam.2020.113034.

[2] G. Wu, Y. Wei, *A Power-Arnoldi algorithm for computing pagerank*, Numer. Linear Algebra Appl. (14), 2007, 521–546, http://refhub.elsevier.com/S0377-0427(20)30325-3/sb29.

[3] Golub, G., Greif, C., *An Arnoldi-type algorithm for computing page rank*, Bit Numer Math 46, 759–771, 2006, https://doi.org/10.1007/s10543-006-0091-y

[4] Amy N. Langville and Carl D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, USA, 2006, http://refhub.elsevier.com/S0377-0427(20)30325-3/sb40.

[5] T. Haveliwala, S. Kamvar, *The Second Eigenvalue of the Google Matrix*, Technical report, Stanford University, Stanford, CA, 2003, http://refhub.elsevier.com/S0377-0427(20)30325-3/sb41.

[6] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1992, http://refhub.elsevier.com/S0377-0427(20)30325-3/sb42.

[7] K. Wu, H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl. (22), 2000, 602–616, http://refhub.elsevier.com/S0377-0427(20)30325-3/sb38.

[8] Xueyuan Tan, *A new extrapolation method for PageRank computations*, Journal of Computational and Applied Mathematics, Volume 313, 2017, Pages 383-392, ISSN 0377-0427, https://doi.org/10.1016/j.cam.2016.08.034.

[9] Ronald B. Morgan, Min Zeng, *A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity*, Linear Algebra and its Applications, Volume 415, Issue 1, 2006, Pages 96-113, ISSN 0024-3795, https://doi.org/10.1016/j.laa.2005.07.024.

[10] James W. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1996, https://doi.org/10.1137/1.9781611971446.